
Tag Clouds in Software Visualisation



Jessica Emerson

Department of Computer Science and Software Engineering

University of Canterbury

A thesis submitted in partial fulfilment
of the requirements for the degree of

Master of Science

2014

Abstract

Developing and maintaining software is a difficult task, and finding effective methods of understanding software is more necessary now than ever with the last few decades seeing a dramatic climb in the scale of software. Appropriate visualisations may enable greater understanding of the datasets we deal with in software engineering. As an aid for sense-making, visualisation is widely used in daily life (through graphics such as weather maps and road signs), as well as in other research domains, and is thought to be exceedingly beneficial. Unfortunately, there has not been widespread use of the multitude of techniques which have proposed for the software engineering domain.

Tag clouds are a simple, text-based visualisation commonly found on the internet. Typically, implementations of tag clouds have not included rich interactive features which are necessary for data exploration. In this thesis, I introduce design considerations and a task set for enabling interaction in a tag cloud visualisation system. These considerations are based on an analysis of challenges in visualising software engineering data, and the perceptive influences of visual properties available in tag clouds.

The design and implementation of interactive system *Taggle* based on these considerations is also presented, along with its broad-based evaluation. Evaluation approaches were informed by a systematic mapping study of previous tag cloud evaluation, providing an overview of existing research in the domain. The design of Taggle was improved following a heuristic evaluation by domain experts. Subsequent evaluations were divided into two parts — experiments focused on the tag cloud visualisation technique itself, and a task-based approach focused on the whole interactive system. As evidenced in the series of evaluative studies, the enhanced tag cloud features incorporated into Taggle enabled faster visual search response time, and the system could be used with minimal training to discover relevant information about an unknown software engineering dataset.

Contents

Contents	ii
1 Introduction	1
1.1 Motivation	2
1.2 Research questions	3
1.3 Approach	5
1.4 Contributions	6
1.5 Outline	8
2 Background	11
2.1 The problem with software development	11
2.1.1 Iterative development methodologies	12
2.1.2 Automated unit testing	12
2.1.3 Project management tools	13
2.1.4 Design patterns	13
2.1.5 Code smells	13
2.1.6 Metrics	14
2.1.7 Static and dynamic code analysis	15
2.1.8 Refactoring	15
2.2 Visualisation	16
2.2.1 Perception and cognition	17
2.2.2 Visual search	17
2.2.3 Graphical representation	18
2.2.4 General techniques	18

2.2.5	Interaction techniques	19
2.3	Software visualisation	21
2.3.1	What can we visualise?	22
2.3.2	Quality assurance	22
2.3.3	Process management	23
2.3.4	Architecture	23
2.3.5	Software evolution	24
2.3.6	Runtime data	25
2.4	Tag clouds	25
2.4.1	Why tag clouds?	27
2.4.2	Tag clouds in software engineering	27
2.4.3	Sourcecloud	28
2.4.4	Sonar platform	29
2.5	Summary and discussion	30
3	Design Considerations for Interactive Tag Cloud Visualisation	31
3.1	Visual variables	32
3.1.1	Visual mapping	32
3.1.2	Layout and order	35
3.1.3	Tag length and position	35
3.1.4	Font size	36
3.1.5	Font Family	37
3.1.6	Font colour	37
3.1.7	Background Colour	39
3.1.8	Font Style	39
3.2	Challenges in software visualisation	41
3.3	Datasets for the tag cloud technique	42
3.4	Task types to enable data exploration in a tag cloud	45
3.5	Summary and discussion	46
4	Taggle: A Tag Cloud Visualisation Tool	49
4.1	Original prototype	50
4.2	Data model and transformation	51

4.3	Visual encoding and mapping selection	52
4.3.1	Categorical and continuous data	53
4.3.2	Tag	55
4.3.3	Order	55
4.3.4	Size	56
4.3.5	Colour and transparency	56
4.4	Software engineering tasks	58
4.4.1	Filtering textual data	60
4.4.2	Dealing with large scale data	62
4.4.3	Identify similar characteristics of data	64
4.4.4	Identifying data distribution	64
4.4.5	Identify data correlations and outliers	66
4.4.6	Finding minimum/maximum values	67
4.4.7	Comparison of data elements	68
4.5	Summary and discussion	69
5	Systematic Mapping Study of Tag Cloud Research	72
5.1	Strategies for evaluation	73
5.2	Systematic mapping study	74
5.3	Methods	75
5.3.1	Data sources and search strategy	75
5.3.2	Primary study selection	76
5.3.3	Data extraction	78
5.4	Results	81
5.4.1	Research topic	82
5.4.2	Research approach and methods	84
5.4.3	Visualisation domain	85
5.5	Summary and discussion	85
5.6	Conclusion	88
6	Evaluation Strategy for Taggle	90
6.1	Overall evaluation strategy	91
6.2	Selected evaluations	93

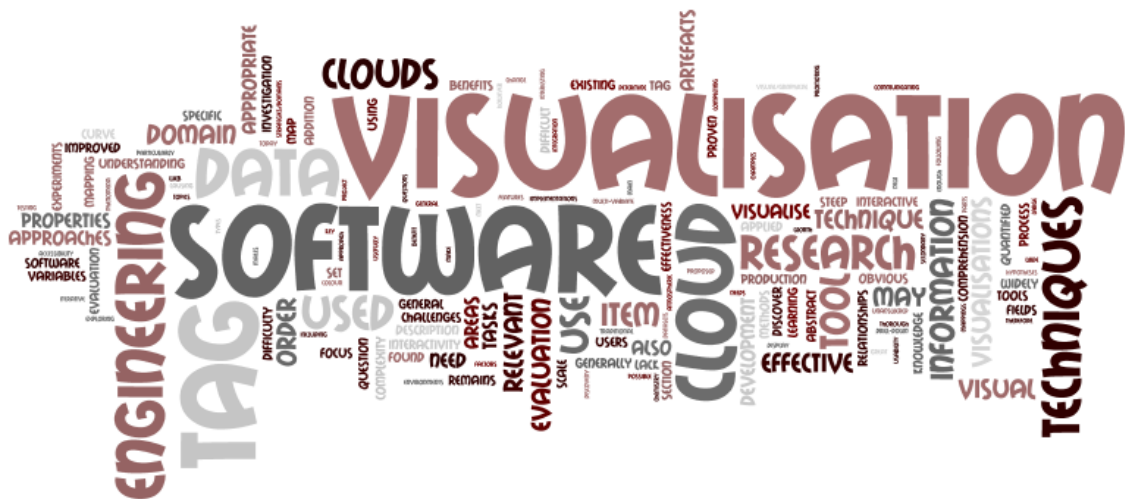
6.3	Dataset generation	95
6.4	Conducting an eye-tracking experiment	96
6.5	Summary and discussion	97
7	Heuristic Evaluation of Taggle	99
7.1	Heuristics	100
7.2	Participants	101
7.3	Apparatus	101
7.4	Tag corpora	101
7.5	Procedure	101
7.6	Results	104
7.6.1	Heuristics — issues, comments and observations	104
7.6.2	Heuristics — guideline checklist	107
7.6.3	Questionnaire	112
7.7	Adaptations resulting from the evaluation	117
7.8	Summary and discussion	120
8	Experiment One: Tag Colour Placement	122
8.1	Foreground or background colour	123
8.2	Participants	124
8.3	Apparatus	124
8.4	Tag corpora	124
8.5	Procedure	124
8.6	Design	125
8.7	Task	126
8.8	Measurements	127
8.9	Hypotheses	127
8.10	Results	127
8.10.1	Eye-gaze data analysis	127
8.10.2	Response time	128
8.10.3	Significance testing	133
8.11	Threats to validity	134
8.12	Summary and discussion	135

8.13	Conclusion and future work	136
9	Experiment Two: Dual Mappings	137
9.1	Single or dual data mappings	138
9.2	Participants	138
9.3	Apparatus	139
9.4	Tag corpora	139
9.5	Procedure	139
9.6	Design	139
9.7	Task	140
9.8	Measurements	140
9.9	Hypotheses	140
9.10	Results	142
9.10.1	Eye-gaze data analysis	142
9.10.2	Response time	142
9.10.3	Significance testing	149
9.11	Threats to validity	149
9.12	Summary and discussion	150
9.13	Conclusion and future work	151
10	Experiment Three: Knowledge Discovery	152
10.1	Knowledge discovery support in Taggle	152
10.2	Participants	153
10.3	Apparatus	153
10.4	Tag corpus	154
10.5	Procedure	155
10.6	Tasks	155
10.7	Measurements	157
10.8	Results	157
10.8.1	Eye-gaze and mouse-tracking data analysis	157
10.8.2	Task analysis	164
10.9	Workload measures	167
10.10	Summary and discussion	168

10.11 Conclusion and future work	170
11 Conclusions	171
11.1 Review of thesis contributions	172
11.2 Limitations and future work	174
11.3 Closing remarks	176
Appendices	177
A Software engineering data	178
A.1 Eclipse metrics plugins	178
A.2 Software analysis platforms	178
A.3 Metric frameworks/tools	179
B Heuristic artefacts	181
B.1 Heuristic descriptions	181
B.2 Heuristic checklist	187
B.3 Typical usage patterns	188
B.4 Usability issues found linked to heuristics	191
C Eye-gaze visualisations	193
D Publications	201
References	216

1

Introduction



1.1 Motivation

Visualisations, or graphical representations of data, are thought to be an effective method of communicating information and are used everywhere in our daily lives, from weather maps to road signs. Visualisations based on an underlying physical model are heavily used in scientific fields such as physics, chemistry, medicine and atmospheric sciences to increase understanding and to confirm or reject hypotheses. In other research areas, information visualisation is used to discover interesting phenomena in unknown and abstract data. Visualisation has long been used as a way to deal with large datasets — the more data in a dataset, the greater the need and the greater the pay-off for visualisation. Ultimately all visualisation is about “things”, their detailed properties and their relationships. In tag clouds, “things” are displayed through tags, properties are displayed through various field values, and relationships are shown through the tag order, layout, or through explicit connections.

Software is a domain where visualisation is sorely needed. Modern software systems are notoriously large and complex — this complexity makes them difficult to understand, develop and maintain, causing costly IT project failure, and budget or timeline blow-outs. We clearly need more effective methods of promoting comprehension of software in addition to the modern software development approaches such as iterative development and automated unit testing. Approaches exist to visualise abstract software artefact properties such as algorithms, metrics, process data and relationships. However, despite the seemingly obvious benefits of using visualisation and the existing research efforts surrounding the software visualisation domain, it is not widely practised or integrated into mainstream development environments [Reiss, 2005].

The question of why this is the case remains largely unanswered, but some possibilities include; a) existing models and techniques having a steep learning curve, or the concepts are difficult to quickly grasp, b) visualisations not scaling well to the demands of a software engineering dataset, and c) techniques not being adequately and demonstrably proven to have a benefit, and therefore not potentially being worth the effort of integration into a workflow. Although a wide range of visualisations have been proposed for the software engineering domain,

there remains the need to explore new techniques, particularly those with a low level of conceptual complexity, and for the effectiveness of visualisation tools and techniques to be quantified.

1.2 Research questions

It may be that the difficulty of software comprehension could be improved with effective use of visualisation techniques, but these techniques are not widely applied in industry. A variety of factors including the steep learning curve of visualisation techniques and a general lack of quantified effectiveness potentially contribute to this.

Enter tag clouds, a simple and highly recognisable visualisation commonplace on the web today. Some of the main benefits of using tag clouds as an information visualisation technique are their accessibility and visual interestingness, and the fact they have a set of visual properties (such as font size and text colour) that make it possible to map to data variables. Labels and textual identifiers are an intrinsic part of the visual encoding, enabling users to determine key information without the need to navigate around or drill-down. As an effective visualisation, tag clouds suffer from some drawbacks — these are generally related to the limited interactivity provided in traditional implementations, such as the inability to change order or layouts on the fly. This lack of interactivity causes great difficulty in the process of exploration of data. With the addition of such features generally found in a visualisation tool, can the tag cloud metaphor be extended to successfully visualise multi-variate data such as that found in software engineering?

The research reported in this thesis takes the tag cloud visualisation technique and applies it to the software engineering domain. The primary research question pursued was to discover if visualisation of relevant software engineering data artefacts with a tag cloud could promote a greater understanding of a software system, in particular if it would assist users in completing specific types of tasks. This was divided broadly into the following secondary areas of research:

Design choices for an interactive tag cloud tool

- defining visual properties that may influence perception in tag clouds
- selecting visual properties that are appropriate for data mapping
- how to appropriately use characteristics of visual properties
- challenges and special needs in a software engineering dataset
- task types to enable data exploration in a tag cloud
- applying the resulting design considerations to an interactive tool

Previous evaluations in tag cloud research

- using prior tag cloud evaluations to shape our research focus and build an overview of tag cloud knowledge in general
- the extent to which each topic has been evaluated
- evaluation approaches and methods, fields and domains

Evaluation of our interactive tag cloud tool

- evaluation strategies and methodologies which match our research goals
- how to evaluate our tool in as broader manner as possible
- discovering whether our tool is usable and comprehensible
- discovering whether the tool is easy to learn
- investigating if the enhanced tag cloud techniques provided in our prototype provide an improvement in user performance
- investigating if the tool can be used to discover knowledge about an unknown software engineering dataset, with a minimum of training

1.3 Approach

Using the analysis of design considerations and task types detailed in Chapter 3, we extended and altered existing software to produce a novel interactive tag cloud visualisation tool. This tool was designed to cope with the specific challenges of software engineering datasets using appropriate visual mappings available in tag clouds to render the data, and is introduced in Chapter 4.

It can be a challenging task to evaluate information visualisation techniques because of the difficulties in capturing and quantifying the data exploration process. We embarked on a systematic mapping study (Chapter 5) of previous research evaluating the tag cloud technique or interactive tools that included tag cloud visualisation. Topics, fields or domains that had not been extensively researched, and approaches and methods which had been used for evaluation were identified. This provided a big picture view of what was known about tag clouds, and helped us plan and focus our overall evaluation strategy. An evaluation map (Chapter 6) was created to plan a broad-base investigation of points of relevance for both the tag cloud technique and our interactive tool, with selection of experimental methodology based on research goals.

As part of our overall evaluation strategy, a heuristic evaluation by domain experts was performed (Chapter 7). Subjective user feedback was elicited to clarify research questions around the comprehensibility of the visualisation technique and data mapping process, as well as assessing the system usability. This evaluation generated various prototype design refinements and satisfied us that the tool was mature enough for use in more detailed experimentation.

Following the heuristic evaluation, three experiments were completed in order to explore the potentials and limitations of the interactive tag cloud visualisation tool. In order to obtain a broad evaluation of relevant parts of the tool, these experiments were conducted in both areas of *visualisation use* (the tag cloud technique) and *data analysis process* (a whole-tool approach focused on the knowledge discovery process). Properties of the enhanced tag cloud features utilised in the interactive system were investigated in two experiments using eye-tracking technology (Chapter 8 and Chapter 9) to discover if improvements could be made in user performance for visual search tasks. The results of the *visualisation use*

experiments have wider implications than just for our interactive tool alone, and are relevant for designers of tag cloud visualisations. To evaluate the tool in a more holistic fashion, an empirical user study was conducted to examine data exploration and knowledge discovery support for software engineering data in our interactive system. The extent to which the system was efficient in facilitating knowledge discovery was gauged through analysis of domain appropriate benchmark task completion rates. Eye-gaze data was also collected for all experiments and analysed for user visual search patterns in tag clouds, and to study usage of various areas of interest within the interactive interface.

1.4 Contributions

The contributions of this thesis can be categorised into four areas:

- Design considerations for an interactive tag cloud visualisation system. We analysed the challenges in visualising multi-variate data, and the capabilities of tag clouds — in particular the effects on user perception for available visual properties according to various design principles, guidelines, and current research. We defined the appropriateness of visual variables for data mapping, and task types that should be supported in order to enable data exploration.
- Systematic mapping study of existing tag cloud evaluation research. This provided an overview of existing evaluations of the tag cloud visualisation and tools which incorporated the technique. There was a strong prevalence in the research to focus on web and user generated data domains, using a limited range of evaluation approaches.
- The design and evaluation of a novel tag cloud visualisation system.
 - (a) ‘Taggle’ system created in accordance with the design considerations (through extensions and alterations to existing software) in order to explore multi-variate data such as software quality assurance measurements.

-
- (b) Subsequent design refinements were necessary following a heuristic evaluation by domain experts, where the system was evaluated for usability and appropriateness of exploration of multi-variate data. The study revealed the tag cloud technique of contrasting visual font properties mapped to data fields was felt to be instinctively comprehensible, but the amount of information that could be inferred was greatly dependent on the software’s support for selection of appropriate mappings.
 - (c) The system’s support for data exploration and knowledge discovery was evaluated through an empirical user study. Results were encouraging, showing ‘Taggle’ could be used with minimal training to discover relevant information about an unknown software engineering dataset. Eye-gaze data showed participants who successfully completed tasks highly utilised the data summary panel and rich interactive features such as mapping multiple visual properties to data fields, and static and dynamic filtering.
- The evaluation of enhanced tag cloud features utilised in the interactive system through user experiments. The results of these experiments have implications for designers of tag cloud visualisations, and provide insight into tag cloud visual search patterns.
 - (a) *Tag background colour* Results indicated usage of tag background colour as a data variable field can produce faster visual search response time than font colour in a tag cloud, when the target tag is small.
 - (b) *Dual data mappings* Dual mappings of font size and colour can produce faster visual search response times than singular mappings of font size or colour alone.
 - (c) *Visual search patterns* Previous eye-tracking studies have identified user serial scanning and chaotic search methods within tag clouds. Our eye-tracking data analysis showed the introduction of a visual property hint when performing a search task, can alter the search strategy to the eye-scan path focusing on tags with the target mapping

(efficient feature search). When task complexity is increased, users generally employed combination visual search methods in a tag cloud, switching between visual feature search, serial scanning and chaotic search methods.

1.5 Outline

This thesis presents the following relevant information:

Chapter 2 *Background* Describing current software engineering practices, how they attempt to address quality issues and manage the manifold complexities existing in today’s software systems. Basics in visualisation and software visualisation. Introduction to the tag cloud, benefits of usage, and limitations of currently available software engineering tools which incorporate tag clouds.

Chapter 3 *Design considerations for interactive tag cloud visualisation* Discussion of visual variables in a tag cloud which may be manipulated to represent data variables. Challenges in software visualisation and task types that represent meaningful ways users may interact with software data.

Chapter 4 *Taggle: A tag cloud visualisation tool* Presentation of the tag cloud visualisation tool Taggle implemented according the the design considerations. Description of the data model and transformations, the visual encoding and mapping selection. Examples of how to use the tool to complete the software engineering tasks outlined in the previous chapter.

Chapter 5 *Systematic mapping study for tag cloud research* Strategies that can be employed when evaluating an information visualisation tool. Description of a systematic mapping study performed to identify topics, fields or domains where tag cloud visualisation tools have been evaluated. Discovering what evaluation approaches and methods were used.

Chapter 6 *Evaluation strategy for Taggle* Based on the outcomes of the systematic study, this chapter details the creation of an evaluation map

outlining potential areas where Taggle could be evaluated along with sample methodologies. This was used to strategically plan a series of evaluations covering a broad base of relevant topics. Generation of experimental datasets and how the eye-tracking experimentation was conducted is also discussed.

Chapter 7 *Heuristic evaluation of Taggle* Description of a heuristic evaluation performed to elicit user feedback into the general usability of the tool, and to clarify research questions pertinent to the future experimentation as well as checking the tool was sufficiently mature to use in experimentation. Results and adaptations resulting from the evaluation are presented.

Chapter 8 *Experiment one: tag colour placement* Description of an eye-tracking experiment performed to examine the specifics of the tag cloud technique used in Taggle — specifically to ascertain how altering tag colour placement between font or tag background affected user performance for search tasks. Experiment design, procedure, results including eye-tracking and statistical analyses, and conclusions are presented.

Chapter 9 *Experiment two: dual mappings* Description of an eye-tracking experiment performed to examine the specifics of the tag cloud technique used in Taggle — specifically to ascertain how mapping a data variable to size or colour, compared to mapping a data variable to size and colour together, affected user performance for search tasks. Experiment design, procedure, results including eye-tracking and statistical analyses, and conclusions are presented.

Chapter 10 *Experiment three: knowledge discovery* Description of an eye-tracking experiment performed to examine data exploration and knowledge discovery support in Taggle — specifically to ascertain if and how Taggle supported the discovery of relevant software engineering information and if the tool was sufficiently easy to learn to complete the tasks with minimal training. Experiment design, procedure, results including eye-tracking and statistical analyses, and conclusions are presented.

Chapter 11 *Conclusions* Review of thesis contributions. Future work and limitations of the research are discussed.

Chapter D *Publications* Articles published as a result of this research.

2

Background

A diverse corpus of work in software engineering and visualisation is relevant to the design of an interactive tag cloud visualisation tool for software engineering. This chapter reviews the background work which has shaped this thesis. The problematic issues in software development are outlined in §2.1, along with modern ways to tackle aspects of scale and complexity. Basics in visualisation are discussed in §2.2 and visualisation of software artefacts are detailed in §2.3. Finally, the tag cloud technique is introduced and related tools critiqued in §2.4.

2.1 The problem with software development

The inherent scale and complexity of software has increased dramatically over the last few decades. Contributors to this complexity include software features such as graphical user interfaces and other layers of conceptualisation and abstraction that did not exist in earlier programming languages. The size of software has also increased. It's not uncommon for a system to contain millions of lines of

code and beyond — Microsoft Windows operating system Vista reputedly has over 50 million. Additionally, there is a lack of common hierarchy and structure in software. Complex relationships exist between components, particularly in large scale enterprise applications such as those found in banking and other industries. Software is constantly evolving, due to such things as requirements changing, technology or infrastructure upgrades, bug fixing, and functionality improvements. Developing and maintaining software is a tricky business, therefore it is imperative we have effective methods of promoting the comprehension of software. A number of software engineering practices have been devised to combat the issues of scale and complexity in software development. In an effort to replicate procedures from traditional engineering disciplines, these practices include measurement, analysis and interpretation of results.

2.1.1 Iterative development methodologies

Various development methodologies plan and control the software development process. The waterfall model of development proposed producing requirements analysis and detailed design documents up-front. This was refined in the spiral model which combined iterative development with the structured design process of the waterfall model to allow a more flexible approach. Later, agile development methodologies (inspired by the Agile Manifesto [Beck et al., 2001]) such as SCRUM and XP were devised which promote a more adaptive and flexible approach to design and development. Agile and iterative methodologies are specifically designed to cope with the constantly evolving nature of software through development in small increments. Development methodologies and their process metrics allow the state and transition of a project to be measured.

2.1.2 Automated unit testing

The increased size of software means more code to test and a greater number of defects to find. Automated unit testing using such tools as jUnit¹ and NUnit² mean contracts of an interface can be tested automatically. Unit testing is a

¹<http://www.junit.org/>

²<http://www.nunit.org/>

key feature of Agile methodologies such as Test Driven Development, where it is expected that a test for a given module of code will be written before the module itself is produced. Modules are not considered complete before the unit test has passed. This practice has been shown in empirical studies to produce a significant increase (more than double) in code quality than products developed without using TDD [Bhat and Nagappan, 2006]. Through automated unit testing we are able to measure software fault density, location and severity. We can also measure the quality and adequacy of the testing.

2.1.3 Project management tools

Project management and comprehension tools such as Maven¹ begin to solve some of the issues around a lack of common software structure. There are still limitations though, complex relationships found between componentry remain, and use of Maven binds you to the Java platform. Additionally, introducing a commonly understood structure and build tool into existing legacy code can be an expensive and time consuming business.

2.1.4 Design patterns

Design patterns may add a recognisable structure to software, providing solutions to common software engineering problems. Introduced by the “Gang of Four” in their quintessential guide [Gamma et al., 1994], design patterns exist to solve problems such as application interaction, integration, enterprise, and domain model analysis, amongst others. Anti-patterns can also be identified — these are patterns in software that are commonly used but may be ineffective or damaging.

2.1.5 Code smells

Code smells, a term coined by Kent Beck and popularised in Fowler [1999], are also certain patterns within software, generally categorised into areas of structure, relationship or inheritance. The presence of a “bad” code smell indicates there is

¹<http://maven.apache.org/>

a possibility of a design flaw and the code would benefit from being restructured. There is said to be a certain degree of intuition required in identifying code smells, “you have to develop your own sense of how many instance variables are too many instance variables and how many lines of code in a method are too many lines” [Fowler, 1999, pg. 75].

2.1.6 Metrics

The software engineering community has defined various metrics of software with the purpose of producing quantifiable measurements. Comprehensive details on various software metrics and their calculations and usages may be found in Fenton and Pfleeger [1998] and Henderson-Sellers [1995] and may be grouped into various categories such as:

Size Measuring size of counting attributes. Examples of this kind of metric are LOC, SLOC, KLOC or Halstead software science.

Complexity Measuring the complexity of flow and data control structures such as cyclomatic complexity and NPATH.

Object-oriented metrics Metrics computing complexity for object-oriented languages. The most common example of this is the Chidamber and Kemerer suite.

Quality Metrics calculating intrinsic software quality. Examples include defect density and MTTF.

Process Measuring the effectiveness of the software process. Examples include defects reported by end-users, human effort and calendar time expended.

Metrics are measurements of specific elements of software entities and are used to summarise software and detect outliers in large volumes of data. The measurements may then be used to make informed decisions about the software, to improve its overall quality and determine the progression of specific projects. In practice there has not been widespread adoption of metric use. There is also no standardisation of metric calculations and this can lead to challenges as different results for the same metric can be calculated from different tools.

2.1.7 Static and dynamic code analysis

Dynamic software analysis involves the collection of data during the execution of the system. This kind of analysis has several different motivations: runtime debuggers allow you to step through a running process, code profilers allow you to find performance hotspots, code coverage tools measure the amount of software that has been executed, and memory analysis tools track the memory usage of the software in order to locate leaks and reduce memory consumption. An issue with dynamic analysis tools is they are used late in the development cycle where it is generally considered to be more costly to fix a defect. Static analysis deals with the structure and development of the source code and may be performed much earlier in the development cycle. This analysis includes calculating source-code metrics, using defined patterns to detect potential bugs, and finding instances where coding conventions and rules were broken. An advantage of static code analysis is that defects may be found in parts of code not executed during normal program operation such as error handling routines, or issues involving memory corruption or leaked system resources. (See [Appendix A](#) for a list of static and dynamic analysis tools by type).

2.1.8 Refactoring

Bad code smells, poor metric results, anti-pattern identification, and static or dynamic analysis can be used when trying to determine where and how much code should be refactored in a project. Refactoring is a modern process of rewriting software, without changing the functionality, in order to improve it. It can be described as an ongoing refinement process, which should be evoked as software evolves over time — this is at odds with earlier styles of software development where changes were avoided because of fearing unintended consequences. With the advent of automated unit testing, bugs are less likely to be introduced during the refactoring processing. Refactoring is typically performed to allow greater understandability and extensibility of the code base, in order to simplify maintenance and provide a better platform for ongoing future development. Examples of refactoring techniques in areas of improving data organisation, simplifying conditional expressions, improving abstraction, and breaking code into more logical

pieces can be found in the 1999 classic refactoring reference *Refactoring: Improving the Design of Existing Code* [Fowler, 1999], and on Martin Fowler’s website¹.

2.2 Visualisation

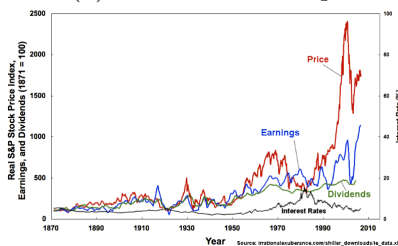
It is said that “a picture is worth a thousand words” and we see this adage embodied in our daily lives — information visualisation and graphics are used everywhere and are considered exceedingly beneficial (see Figure 2.1 for common examples). Interactive visualisation tools are used in various research disciplines to find interesting phenomena in unknown and abstract data. This differs from scientific visualisation which is primarily physically based and used heavily in fields such as physics, chemistry, medicine and atmospheric sciences (see Figure 2.2 on the next page). The following sections present some basic information on visualisation — for more details, refer to [Spence, 2007; Ware, 2004].



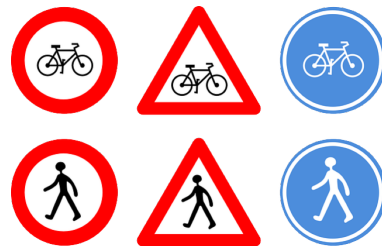
(a) London tube map



(b) Synoptic chart of weather in USA



(c) Plot of S&P stock data



(d) Bicycle and walking road signs

Figure 2.1: Visualisation examples (a)[Lars, 2007] (b)[Frothy, 2008] (c)[NASA, 2006] (d)[OpenClips, 2013]

¹<http://martinfowler.com/refactoring/catalog/index.html>

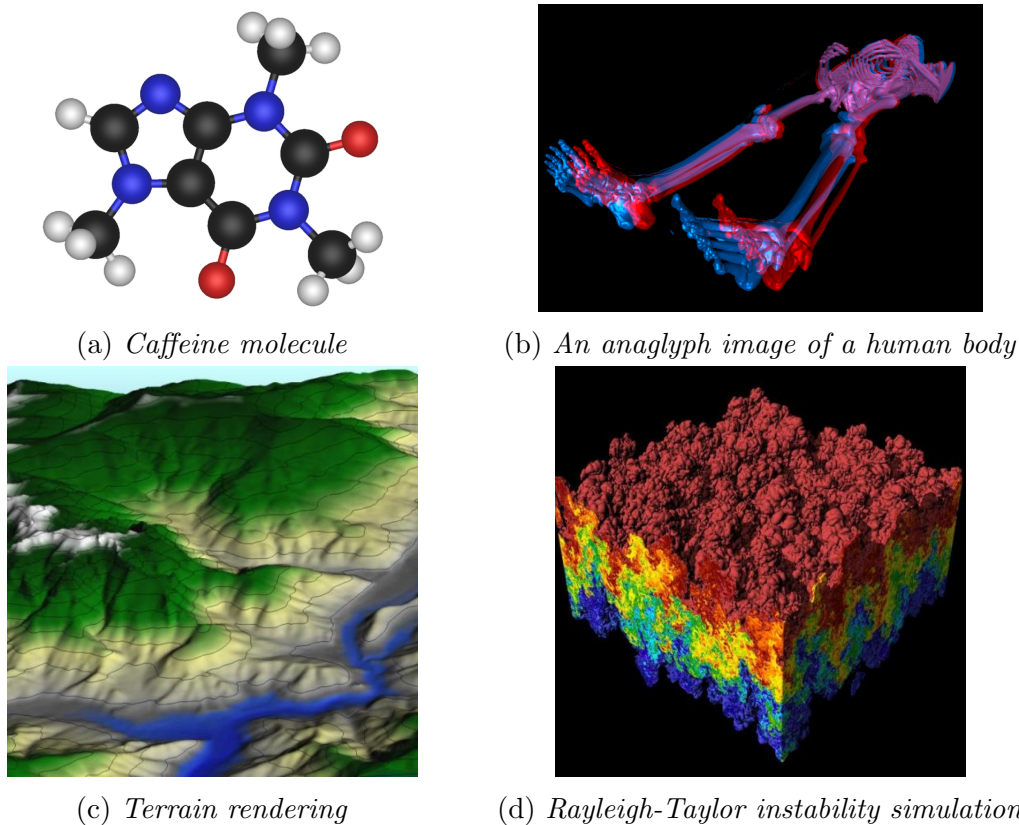


Figure 2.2: *Scientific visualisation* (a)[Stroeck, 2006] (b)[Krueger, 2009]
(c)[Laboratory, 2009] (d)[UCRL, 2007]

2.2.1 Perception and cognition

Cognition is a group of mental processes that includes attention, memory, learning, reasoning, and decision making. Perception is the processing of sensory information and is therefore a part of human cognition. Humans have a well developed sense of sight — we perceive 75 percent of real world information visually. Visualisation is effective because it takes advantage of the brain’s abilities, allowing us to gain insight more intuitively rather than through conscious thinking.

2.2.2 Visual search

Visual search is a perceptual task requiring attention that involves the viewer actively scanning a visualisation or visual environment for a certain object or

feature (the target) among other objects. There are various factors that can influence search performance. Basic attributes that can be considered to guide the deployment of attention (with a strong likelihood of supporting efficient search) are colour, motion, orientation and size [Wolfe and Horowitz, 2004].

2.2.3 Graphical representation

Visualisations are built from shapes and lines which have various properties such as size, length, width, height, volume, position, and colour. Additionally, these primitives can have dynamics that change over time, for example blinking or flashing. These visual properties are what is used to encode the information in a visualisation.

Visualisations can also consist of text (tag clouds introduced in §2.4 are an extreme example of this — other visualisations may use text in passing, such as labelling an icon). Text can be manipulated to encode information through the use of typographical elements such as colour, font face and font styles.

2.2.4 General techniques

There are a great number of visualisation techniques that can be used. Keim and Kriegel [1996] classified visualisation techniques according to their display mode:

1. *Pixel-oriented techniques.* The arrangement of pixels, each dimension value mapped to a coloured pixel, grouped into adjacent areas. Pixel displays generally use one pixel per data value, so this technique can allow visualisation of large amounts of data depending on the display resolution. Appropriate arrangement of pixels can provide information on correlations and dependencies.
2. *Geometric projection techniques.* Geometrically transformed visualisations such as scatterplot matrices and parallel coordinates aim to show interesting properties of multi-dimensional datasets. In a parallel coordinate visualisation, dimensional spaces are mapped onto two display dimensions with axes that are parallel to each other. Each data element is depicted by connected line segments which intersect each of the axes (see Figure 2.3).

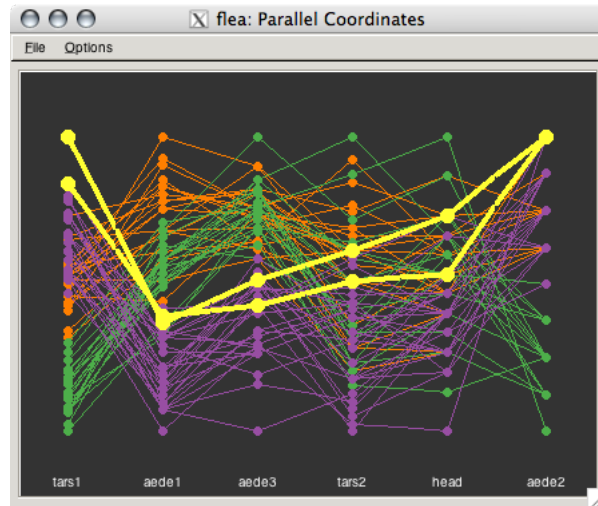


Figure 2.3: Screenshot of GGobi, showing a parallel coordinate plot [Cook, 2008]

3. *Icon-based techniques.* Iconic display methods map multi-dimensional data values to an icon by mapping the attribute values to features of the icon. An example of an iconic display is Chernoff faces [Chernoff, 1973] as shown in Figure 2.4 on the following page.
4. *Hierarchical techniques.* Hierarchies are often drawn as node and edge diagrams, where a node is represented by a shape, and edges are represented by lines. Node and edge diagrams make inefficient use of space, with emptiness at the top left and right. Screen-filling techniques such as treemaps (see Figure 2.6 on page 24) have been developed to fit large hierarchies onto the screen and use space more efficiently.
5. *Graph-based techniques.* Graph drawing facilitates understanding of relationships between objects. Graph-based techniques are used in trees, word graphs, and workflow diagrams. Specific layout algorithms are used to present large graphs.

2.2.5 Interaction techniques

Interaction techniques are features that provide users with the ability to manipulate and interpret visualisations. There are a variety of ways that a user can

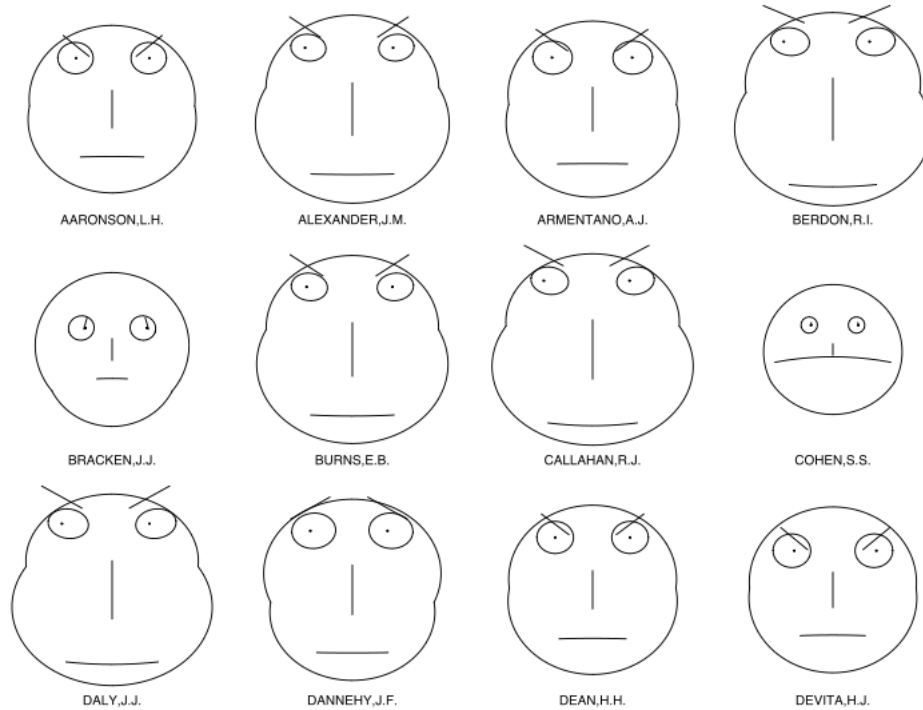


Figure 2.4: *Chernoff faces for evaluations of US judges* [Avenue, 2010]

interact and explore the data, for example:

Focus+Context The underlying premise of Focus+Context is that the user may need both overview and detail information simultaneously. These two types of information can be combined within a single display, through interaction techniques such as fish-eye.

Filtering Uninteresting data elements can be filtered out. Dynamic queries can allow users to control the contents of the display, and focus on items of interest through elimination of other items.

Zoom Items of interest may be zoomed in on. If users wish to know more about a particular area of the data, they can point to this location and click a mouse button until the required level of zooming is achieved.

Brushing and linking Brushing and linking allows multiple visualisations of a dataset to be viewed simultaneously. Brushing of markers within the

visualisations (such as in a scatterplot matrix) can then occur. The brushing and linking process involves selecting one element in a set of visualisations, brushing it with colour, then viewing the other linked visualisations to see the effect.

2.3 Software visualisation

As in information visualisation, in software engineering we want to explore abstract data to find trends and other interesting phenomena. We need help to comprehend and improve existing structures and build new ones. The examples given in Figures 2.1 and 2.2 of information and scientific visualisation were all created with software. Given the success of visualisation in scientific and other research disciplines and the prolific examples of information visualisation and graphics in everyday life, computer generated visualisation and software engineering should be ideally suited. However, software visualisation is not widely practised or integrated into mainstream development environments. An exception to this rule is UML diagrams, which are often used to create system diagrams of class structures and interactions (see Figure 2.5). Even these diagrams, though, are often used for the development of new systems and may be largely ignored after the fact.

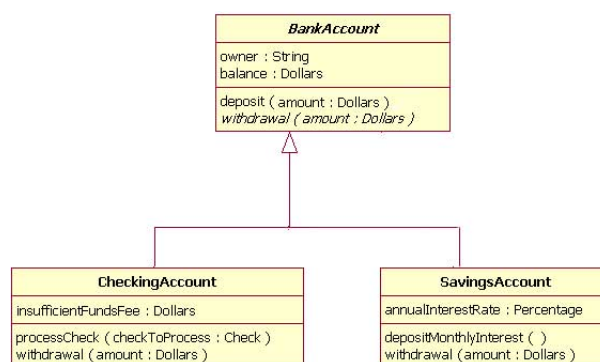


Figure 2.5: *UML Inheritance Diagram* [Bell, 2004]

So what is the reason for the current lack of enthusiasm for software visualisation? Reiss [2005] discusses this conundrum in his paper “The paradox of Software Visualisation”, and outlines various possibilities which include reasons

Table 2.1: *Visualisation of Software Artefacts*

Type	Area
Quality assurance	metrics code smells heuristics
Process management	story/task management time allocation and management anomaly/issue detection project trends user achievement production support
Architecture	algorithms and control flow relationships and hierarchies interactions
Software evolution	source code repositories metrics structures
Runtime data	debugging memory management

such as; lack of scaling to a suitable dataset size, interactive tools not providing answers to specific questions, low development workflow integration, lack of ease of use and high learning time, and neglect to adequately prove benefits. In the following sections, we describe the areas of software development where visualisations have been proposed, and introduce various devised techniques.

2.3.1 What can we visualise?

Visualisation of software artefacts can help us in a number of areas, such as those detailed in Table 2.1. A variety of approaches have been proposed for these areas.

2.3.2 Quality assurance

Visualisation of software quality metric data often involves using a selection of multi-variate data visualisation techniques such as histograms, scatterplots or

parallel coordinates. Kiviat charts (star glyphs) are also a commonly used technique for displaying multi-dimensional data.

Various 3D visualisations based on real-world metaphors like cities and landscapes have been developed. These have been applied in both software quality and architectural areas [such as [Alam and Dugerdil, 2007](#); [Irwin and Churcher, 2003](#)] where metaphors such as 3D virtual worlds and building blocks have been explored.

It is possible to create ambient visualisations using alternate senses such as sound, odour, or vibration. [Murphy-Hill and Black \[2010\]](#) proposed a novel smell detector called Stench Blossom, that provided an interactive ambient visualisation designed to give programmers a high-level overview of the smells in their code. This and other visualisations of code smells such as jCosmo [[Emden and Moonen, 2002](#)] have been integrated in the development environment via a plugin.

2.3.3 Process management

Agile methodologies, used popularly in software project management, are inherently visual. Board views (used for story and task management) and burndown charts (time management, team status progression) are used in many agile teams. Also widely used are Gantt charts (a type of bar chart), which are visualisations incorporated into project management software illustrating a project schedule.

2.3.4 Architecture

Algorithm visualisations for structured programming may be generated with graphical notations Nassi—Shneiderman diagrams (structograms). Nested boxes are used to represent simple statements and program control flow. Control flow graphs, first introduced by Frances E. Allen [[Allen, 1970](#)], show all paths that might be traversed through a program during its execution. These graphs are used in compiler optimisation and static program analysis tools.

Created by Booch, Rumbaugh and Jacobsen, Unified Modelling Language (UML) [[uml, 2007](#)] is a popular and widely used set of graphical notations used to display relationships, interaction models and hierarchies in software. The set is comprised of a large number of diagrams including class/object models, use

cases, behaviour and interaction diagrams, implementation diagrams and model management. These diagrams are most frequently used to visualise architectural elements of software.

Created by Ben Shneiderman in the nineties [Shneiderman, 2009], treemaps can show hierarchical data such as that found in software architecture, through the use of nested rectangles (see Figure 2.6).

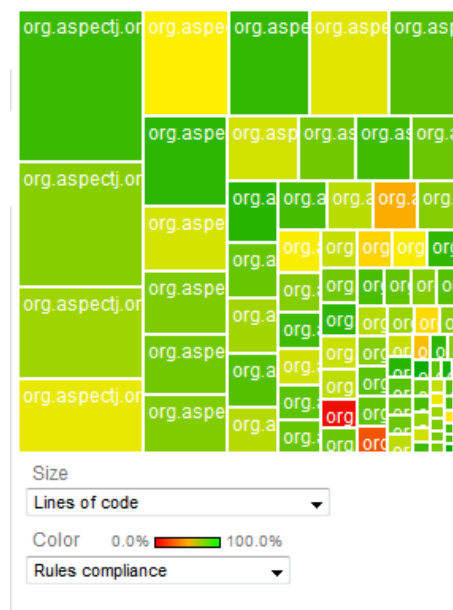


Figure 2.6: *Aspectj Treemap (created by the Sonar Software Analysis Platform)*

2.3.5 Software evolution

Visualisations of source code version histories are produced to analyse the evolution of a software system over time. This can reveal commonalities and irregularities in the development process. An example of this kind of visualisation is a ‘revision tower’ [Taylor and Munro, 2002] which allows people to see active areas of a project, how often changes are made, and how work is shared out.

Some tools produce visualisations of software metrics over time in order to provide a useful picture of software quality trends. One such example of this is the ‘SeeSoft system’ [Eick et al., 1992] which produces a space-filling visualisation for software metrics that are related to individual lines of code. Metrics such as

code age can be viewed colour-coded, so that it is possible to see what parts of the system have recently been touched.

2.3.6 Runtime data

Dynamically generated visualisations to assist the debugging of runtime data are sometimes included as plugins in the development environment, such as the Eclipse Memory Analyser (MAT)¹. These tools may use a collection of visualisation techniques such as histograms, pie-charts and line graphs to illustrate measurements.

2.4 Tag clouds

Tag clouds are a common example of information visualisation found on the World Wide Web and are used to embody text — words, two-word phrases and symbols. These items are grouped together to form a visual representation of the data. Each element of the visualisation is referred to as a tag — this may be website keywords which are hyperlinked to related resources. The frequency of a word (and therefore assumed importance) is highlighted by the font size or colour. This makes the most important and prominent keywords easy to quickly identify as well as showing the relative importance of keywords. They can be displayed in a variety of layouts, most commonly alphabetically (see Figure 2.7 on the next page for a typical example).

An early example of a weighted list of keywords can be found in Douglas Coupland’s 1995 novel *Microserfs*. In this novel, a computer algorithm selects random phrases from an electronic diary creating a set of “subconscious files”. In 2002 Jim Flanagan created a Perl module (Search Referral Zeitgeist), which generated a graphic of website referrers. Based on this implementation, photosharing site Flickr², founded in 2004, created a “tag cloud” visualisation showing tag popularity through font size. Tag clouds then started appearing as a navigation aid on Web 2.0 websites such as Del.icio.us³.

¹<http://www.eclipse.org/mat/>

²<http://www.flickr.com>

³Now known as <http://www.delicious.com/>

2.4.1 Why tag clouds?

Tag clouds initially appear simple. This, along with high exposure online, makes them potentially more accessible to users than visualisations such as treemaps. This apparent simplicity may positively effect ease of use and learning ability, which were identified as important factors to visualisation takeup in the software engineering industry [Reiss, 2005]. Another potential benefit of tag clouds is that they are reportedly perceived by some users to be visually interesting, appealing or otherwise aesthetically pleasing [Hearst and Rosner, 2008]. Some studies [such as Kuo et al., 2007] have also reported a high level of user satisfaction using them.

In tag cloud visualisation, text labels are an intrinsic part of the visual encoding. This is particularly beneficial for software engineering datasets where textual labels are often used as identifiers.

Most current tag cloud implementations support limited interactivity. In order to effectively explore large datasets greater interactivity needs to be incorporated into the tag cloud visualisation interface, such as adding support for Shneiderman’s visual information seeking mantra [Shneiderman, 1996].

2.4.2 Tag clouds in software engineering

In the software engineering domain, industry tools and academic research have not largely embraced tag clouds as a visualisation technique. There has been discussion of tag clouds for visualising relationships and structure [such as Anslow et al., 2008; Bajracharya et al., 2010; Kurtz, 2011] and also for source control evolution [Kuhn and Stocker, 2012]. These examples show some interesting possibilities for inclusion of tag cloud based techniques in software engineering, but more extensive empirical evaluations are needed (of these papers only Kurtz [2011] was included in the systematic mapping study detailed in Chapter 5, as it involved a usability evaluation).

2.4.3 Sourcecloud

In industry there are very few existing software engineering tools which utilise a tag cloud visualisation technique. Eclipse plugin Sourcecloud¹ produces a Wordle-like visualisation of the text within a class, package or project with font size weighted by term frequency and colours assigned arbitrarily. The motivation behind this tool is to give an impression of how easy the code base is to understand by comparing proportions of domain-specific classname tags against core Java API classname tags. We produced a visualisation of opensource project Aspectj² (see Figure 2.9). Some problems are apparent: the use of colour is a distracting factor (as it is not mapped to any variable), so some tags may appear more important than others without reason. An overview of all classes cannot be seen (due to a maximum word parameter), so words we are interested in may not be visible.



Figure 2.9: *AspectJ² source code visualised with Sourcecloud¹*

¹<http://misto.ch/tag/eclipse/>

²<http://www.eclipse.org/aspectj/>

2.4.4 Sonar platform

The Sonar Software Analysis Platform¹, which allows users to explore and visualise software metric data, comes bundled with a tag cloud component (see Figure 2.10). The tag font size is mapped to metric LOC (number of code lines for the class) and the tag colour is mapped to a rules compliance metric. The class name without the package can be problematic if different packages contained identically named classes. On the other hand, inclusion of the package name could dramatically increase the size of the cloud. The cloud produced is already a very large tag cloud, and may require scrolling. On the plus side, it is easy to quickly identify which classes contain a relatively large number of lines (CompletionEngine, CodeStream etc). The Sonar cloud tags are also used for navigation, and are hyperlinked through to pages showing individual metric data for the selected class.

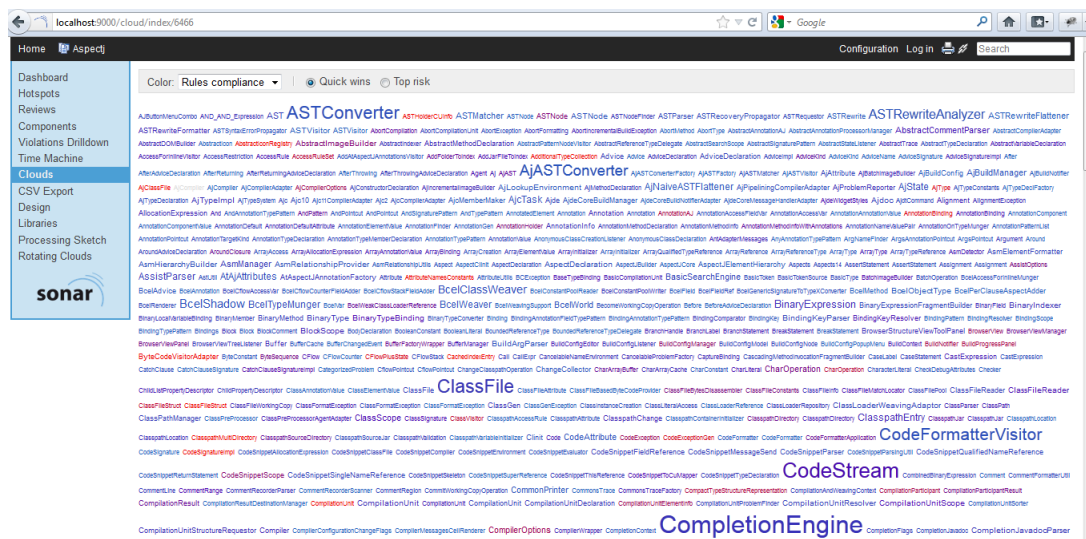


Figure 2.10: *AspectJ² visualised in Sonar*

¹<http://www.sonarsource.org/>

2.5 Summary and discussion

There is a comprehension problem in software development created by the huge size, constantly evolving nature and complex relationships. Modern ways to tackle aspects of scale and complexity include iterative development methodologies, automated unit testing and static or dynamic code analysis tools. Visualisation is a way to handle the reduced, but still overwhelming datasets we need to deal with in software engineering, and is considered useful in other domains (such as scientific research) as well as in general daily life. Despite a plethora of visualisation techniques suggested for the software domain, there remains a lack of widespread use of such techniques.

Tag clouds are a highly recognisable visualisation of low-level complexity which deal primarily with text. Conventional implementations of tag clouds have limited interactivity which creates issues for data exploration. This research is focused on the evaluation of an interactive tag cloud visualisation tool. Can the tag cloud metaphor be extended to successfully visualise multi-variate data such as that found in software engineering? Both the Sourcecloud plugin and the Sonar platform visualisation show it is possible to apply the tag cloud technique directly to source code to identify and explore software quality metrics. Correspondingly, they also show the effectiveness of the technique relies on careful consideration given to various issues such as user perception of visual properties, and rich interactive features to allow data exploration.

3

Design Considerations for Interactive Tag Cloud Visualisation

Tag clouds have been used to visualise books, speeches, and other text through online tools such as Wordle¹, TagCrowd² and Manyeyes³. While these tools easily create aesthetically pleasing clouds for users, their full potential for information visualisation is unrealised. As we saw in §2.4.2, those few tools which have sought to apply the tag cloud paradigm to multi-variate data, have failed to take into consideration certain aspects (such as long identifiers) which have consequently proved problematic. In this chapter we develop design considerations for an interactive tag cloud visualisation system by reviewing design principles, guidelines, and research in general information and tag cloud visualisation. Furthermore, we present a set of task types that should be supported in order to enable data

¹<http://wordle.net/>

²<http://tagcrowd.com/>

³<http://www-958.ibm.com/software/analytics/manyeyes/>

exploration. These task types are generated from analysing the challenges in visualising software and multi-variate data, as well as the capabilities of tag clouds. The task types and design considerations are used to inform the design of our interactive tag cloud visualisation system ‘Taggle’, presented in Chapter 4.

3.1 Visual variables

Tools such as Wordle or TagCrowd generate tag clouds which display word frequency counts through the font size of individual tags. Other visual properties (such as colour, ordering and typeface styles) are generally ignored or used in a decorative fashion. We contend that these alternate visual properties available in tag clouds can and should be used to represent other data variables or reinforce mappings. With these visual properties representing data, a tag cloud can support users in tasks such as gisting, searching and knowledge extraction. In the creation of our interactive tag cloud visualisation system Taggle, consideration had to be given to the amount of influence on user perception for each visual variable, as well as their individual properties which affected suitability of being mapped to data.

Due to the textual nature of tag cloud visualisation, many visual properties in tag clouds relate to font characteristics. Phrase or keyword emphasis in tag clouds are manipulated via typographical techniques. These can be applied with various design principles [such as those outlined in the seminal manual “The elements of typographical style” [Brighurst, 2002](#)]. Tag emphasis in clouds (for individual data points) may be created through manipulation of variables such as size, colour, font family or style. Visual properties of tag clouds that may influence perception can be seen in [Table 3.1 on the following page](#), although not all properties may be suitable to map to data variables. An example of data mapped to some of these properties may be seen in [Figure 3.1 on page 34](#).

3.1.1 Visual mapping

The term ‘visual variables’, as introduced by [Bertin \[1983\]](#) in the “Semiology of Graphics”, refers to a specified set of symbols that can be applied to data in order

Table 3.1: *Visual properties that may influence perception in tag clouds*

Visual Property	Types
Layout	typewriter spiral
Order	alphabetical semantic random
Tag length	variable number of characters equal number of characters
Tag position	top left quadrant top right quadrant bottom left quadrant bottom right quadrant
Font size	
Font family	Serif vs Sans-serif Arial Times New Roman
Font colour	hue saturation value
Background colour	hue saturation value
Font style	bold vs normal italics vs normal all capitals vs mixed case underline vs normal

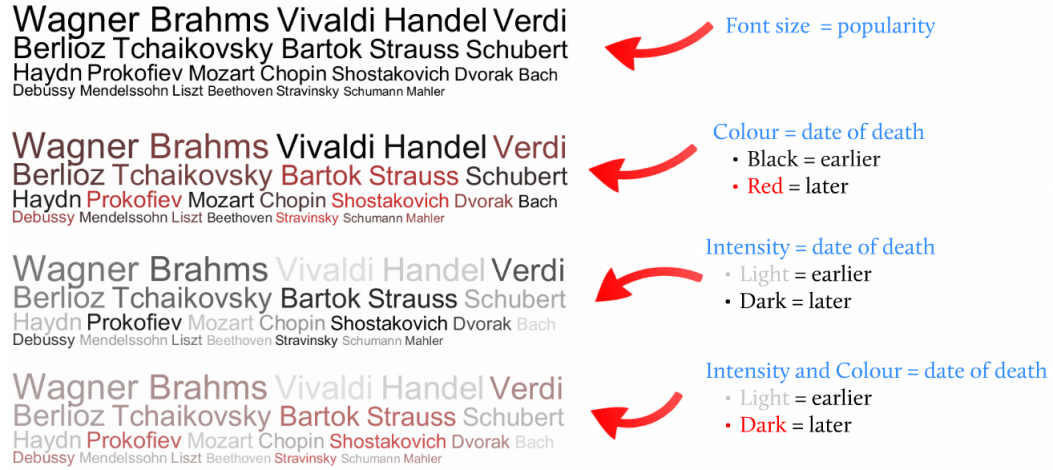


Figure 3.1: *Popularity ranking of composers: mapping of artificial data to tag cloud visual properties (tag clouds generated by Taggle).*

to translate information. This process of mapping data to visual properties is called ‘visual mapping’. The visual variables are defined as position, size, shape, value, colour (hue), orientation, and texture. Choosing a particular variable to map to data depends on an analysis of the characteristics of the variable. Each variable’s characteristics are defined from the following list of perceptual approaches:

- Selective: If a data point can easily be selected as being different from the other data points.
- Associative: If multiple data points can be perceived as being similar.
- Quantitative: If data points can be perceived as being proportional to one another.
- Ordered: If data points can be interpreted in an order.

When creating a data visualisation, it is important to know and appropriately use the characteristics of a visual variable. In the following subsections the effects on user perception for each tag cloud visual variable is discussed with respect to Bertin’s perceptual approaches, other design guidelines and tag cloud research. Likely suitability for data mapping within a tag cloud is determined.

3.1.2 Layout and order

Layouts of tags within a tag cloud are generally based on either a typewriter style (tags arranged from left to right, and top to bottom), or arranged in a spiral pattern. Relevant ordering of the tags within the tag cloud layout (within constraints imposed by such things as window shape) is important for users of the visualisation tools when applying search and locate tasks. Previous research has indicated alphabetical ordering of tags is generally preferable to random or semantically clustered layouts when searching for a named tag within a tag cloud, although semantic clustering can provide improvements over random arrangements [Halvey and Keane, 2007; Schrammel et al., 2009b].

However, the ordering property is also suitable for search and locate tasks using variables not related to the tag text. For instance, using the ordering property to locate tags with maximum or minimum values of a variable, in Figure 3.4 on page 43 once it is understood that the ordering of the tags is related to the popularity of a name within the USA, it is easily found that the top three most popular boys names in the US are Michael, Christopher and Matthew. Likewise, in Figure 3.2 on page 40, it can easily be seen that Bahamas had the greatest number of gold medals in the Olympic medal ranking based on a per-capita ranking system.

- **Tag order:** *Visual property tag order is suitable for mapping to data fields*

3.1.3 Tag length and position

Longer tag lengths have been shown to have an effect on user perception of tag importance [Bateman et al., 2008]. Tags placed in the upper left quadrant of a tag cloud are found more quickly [Bateman et al., 2008] and are also better recalled [Rivadeneira et al., 2007]. Analysis of eye-tracking data has shown the upper left quadrant of a tag cloud receives the most attention [Lohmann et al., 2009; Schrammel et al., 2009a]. It is possible the upper left quadrant dominance is due to western language reading patterns. Due to this quadrant prominence, tag position is a particularly appropriate mapping for search and locate tasks through use of the ordering property (§3.1.2) when tag clouds are arranged in a typewriter fashion.

-
- ***Equal length tag identifiers:*** *It should be possible to set the length of the tag identifiers to an equal length to minimise effects on user perception*

3.1.4 Font size

Text size manipulation can be a very effective way of creating emphasis. Empirical research on tag cloud visual properties has identified size as having a significant effect on user perception [for example Bateman et al., 2008; Halvey and Keane, 2007; Lohmann et al., 2009]. This means large font sized tags are found more quickly than small font sized tags. For example, in Figure 3.5 on page 43 user names ‘pam120’, ‘kfc172’ or ‘gey66’ can immediately be located in the tag cloud due to their prominence from a comparatively large font size.

The size visual variable is selective, associative, ordered, and quantitative [Bertin, 1983], although care must be taken when using size quantitatively, as changes in size from volume or area may be difficult to interpret [Carpendale, 2003]. According to [Bateman et al., 2008; Schrammel et al., 2009a] font size can be accurately compared in a tag cloud.

Because of canvas and screen boundaries, there are limitations in the maximum font sizes which can be displayed within a tag cloud. With regard to minimum font sizes, guidelines based on reading performance research state a 9pt font limit for web pages or screen media [pg 107, chap 11:8 Health and Services, 2006].

For data mapping purposes, font size is an appropriate visual variable candidate for mapping to data variables in a tag cloud. Careful attention should be paid to minimum font sizes for reading ease.

- ***Font size:*** *Visual property tag order is suitable for mapping to data fields*
- ***Constrained font sizes:*** *Font size should be constrained to greater than 9pt, and a suitable maximum font size according to canvas and screen boundaries*
- ***Comparable tags:*** *Tags should be able to be compared by moving closer together to assist quantitative comparisons*

3.1.5 Font Family

Typographical characteristics such as font family were not mentioned by Bertin [1983]. The shape visual variable, which is the most closely related variable mentioned, is not perceived as an especially effective variable. According to Carpendale [2003], shape may be selective and associative, providing there are minimal data points or minimal shape variations.

In tag cloud research, Waldner et al. [2013] found text orientation and shape modifications performed significantly worse than colour coding for distinguishing tag categories. Many users also perceived rotated tags as unstructured and unattractive. Shape differences caused by serifs or font styles were hard to detect in controlled experiments using the Helvetica font style. It is also possible that manipulation of font family may alter user perception of other font styles used as a mapping variable, such as bold or italic. It does not seem that font family would be an effective data mapping visual property within a tag cloud.

Research shows that reading speed is best when users are presented with familiar fonts such as Times New Roman, Arial or Helvetica [pg 106, chap 11:7 Health and Services, 2006]. These fonts may therefore be preferable in a tag cloud for reading accuracy.

- **Font family:** *Visual property tag order is not suitable for mapping to data fields*
- **Familiar fonts:** *For reading accuracy, fonts should be familiar such as Times New Roman, Arial or Helvetica*

3.1.6 Font colour

In a study of the effectiveness of textual retinal properties in tag clouds, Waldner et al. [2013] found that after font size, colour (both as text colour or as the tag's background colour) was the most effective visual text variable for encoding nominal and ordinal data. On the other hand, transparency was disliked by users and lead to inaccurate results when determining tags of relevance. Bateman et al. [2008] found that colour intensity (saturation/transparency) had a relatively good influence on user perception, although not as strong as font size.

Preston et al. [2010] investigated the effectiveness of typographical emphasis techniques (such as colour, bold and italics) on computer presentation software. They found that use of colour in font emphasis techniques generally elicited significantly faster response times identifying text than achromatic techniques such as bold and italic, provided a suitable colour contrast was given.

Colour value has properties selective, associative and ordering [Bertin, 1983], whereas colour hue has only selective and associative properties and cannot be perceived by a viewer in an ordered fashion. Value may be considered to be quantitative also, in that lighter value colours are perceived to be related to smaller numbers and darker colour values to higher numbers, but actual quantitative comparison can be difficult (for example perceiving one shade of colour as being three times darker than another shade).

Ware’s information visualisation guidelines advise not using more than ten colours for coding symbols (especially if the symbols are to be used against a variety of backgrounds) [pg 124, chap 4, G4.15 Ware, 2004]. Ware also recommends twelve specific colours for use in coding: red, green, yellow, black, blue, white, pink, cyan, grey, orange, brown, and purple [pg 126, chap 4, G4.18 Ware, 2004].

For data mapping purposes, it may be more useful to consider colour value and saturation as being aligned to a transparency mapping often employed as a visual property in tag clouds. Both hue and transparency are appropriate candidates for mapping to data variables in a tag cloud. Consideration must be given to colour choice, contrasts and quantitative mapping.

- **Colour hue:** *Visual property colour hue is suitable for mapping to data fields*
- **Colour transparency:** *Visual property colour transparency is suitable for mapping to data fields*
- **Colour selection:** *Colour codes should be taken from Ware’s colour code recommendations (red, green, yellow, black, blue, white, pink, cyan, grey, orange, brown, and purple)*

3.1.7 Background Colour

Background colour is an appropriate data mapping visual variable as an alternative to font colour. As discussed in §3.1.6, [Preston et al. \[2010\]](#) found that use of colour in typographical emphasis techniques elicited faster response times identifying emphasised text than achromatic techniques. This performance improvement was providing a suitable colour contrast was given (such as red, green or blue on a white background). Consideration of text colour against background must be given during colour selection of visual mappings within a tag cloud.

It is possible to manipulate background colour on an individual tag. This could have two possible benefits 1) it allows grouping together of multiple keywords or phrases, and 2) mapping the colour to the background behind the tag may have a greater effect on user perception than mapping the colour to the text. This is because the area of the background behind the tag is greater than the area of the font text itself and in general, the larger the area that is colour coded, the more easily colours can be distinguished [pg 125, chap 4 [Ware, 2004](#)]. Colour coding in the tag background has been found to support more accurate estimation of relevant tags than font colour [[Waldner et al., 2013](#)].

Chapter 6 details empirical research conducted which included investigation of the benefits of background colour manipulation in tag clouds. For an example of text and background colour in a tag cloud see [Figure 3.2 on the next page](#), which shows Olympic medal rankings using both gold first and per-capita ranking systems. Countries which have more than one word in their name are more easily distinguished using background colour mapping.

- **Colour background:** *Visual property colour background is suitable for mapping to data fields*
- **Colour contrasts:** *Strongly contrasting colour schemes should be selected such as red, green or blue on a white background*

3.1.8 Font Style

Font styles or typography in general were not included in Bertin's visual variables [Bertin \[1983\]](#). [Bateman et al. \[2008\]](#)'s exploration of the effects of various prop-

Bahamas Cyprus Denmark Australia
 Estonia Belarus Cuba Croatia
 Czech Republic Azerbaijan Gabon
 Armenia Botswana France Bahrain Finland
 Canada Dominican Republic Bulgaria Belgium
 Colombia Argentina Ethiopia Brazil China Algeria Egypt
 Afghanistan Albania American Samoa Andorra Angola
 Antigua and Barbuda Aruba Austria Bangladesh Barbados Belize Benin
 Bermuda Bhutan Bolivia Bosnia and Herzegovina British Virgin Islands
 Brunei Burkina Faso Burma Burundi Cambodia Cameroon Cape Verde
 Cayman Islands Central African Republic Chad Chile Comoros
 Congo-Brazzaville Cook Islands Costa Rica
 Democratic Republic of Congo Djibouti Dominica East Timor Ecuador
 El Salvador Equatorial Guinea Eritrea Fiji Gambia

(a) *Font colour*

Bahamas Cyprus Denmark Australia
 Estonia Belarus Cuba Croatia
 Czech Republic Azerbaijan Gabon
 Armenia Botswana France Bahrain Finland
 Canada Dominican Republic Bulgaria Belgium
 Colombia Argentina Ethiopia Brazil China Algeria Egypt
 Afghanistan Albania American Samoa Andorra Angola
 Antigua and Barbuda Aruba Austria Bangladesh Barbados Belize Benin
 Bermuda Bhutan Bolivia Bosnia and Herzegovina British Virgin Islands
 Brunei Burkina Faso Burma Burundi Cambodia Cameroon Cape Verde
 Cayman Islands Central African Republic Chad Chile Comoros
 Congo-Brazzaville Cook Islands Costa Rica
 Democratic Republic of Congo Djibouti Dominica East Timor Ecuador
 El Salvador Equatorial Guinea Eritrea Fiji Gambia

(b) *Background colour*

Figure 3.2: *Olympic medal rankings: size and order are mapped to per-capita ranking system. Colour is mapped to standard (gold first) ranking system.*

erties and characteristics of text in tag clouds found that weight (bold style) had a consistently strong influence on user perception. A comprehensive study on the effectiveness of emphasis techniques in presentations by Preston et al. [2010] found bold text performed well with black text on a white background but not with white text on black. Preston et al. [2010] found capitals to be the most effective of the achromatic emphasis techniques, although all four techniques (capitals, bold, italics and underline) did not perform as well as the chromatic techniques (use of colour). Underline and italic emphasis techniques were consistently the least effective means of emphasising text.

Font styles bold and underline are considered more effective as text emphasis than italic or underline. The number of differing values that these typographical styles may show is only two (underline on, underline off — bold on, bold off) so is likely not worth including as a data mapping variable in a tag cloud.

- **Font styles:** *Visual property font styles are not considered suitable for mapping to data fields*

3.2 Challenges in software visualisation

In the software engineering domain, quality metric data distributions are typically heavily skewed and may contain outliers (see Figure 3.3 on the following page). Unlike some other disciplines, where outlying data points may be discarded or ignored, in software engineering these outliers are potentially the most interesting and should be investigated further as potential candidates for refactoring.

In the software visualisation domain, and many datasets in general, there may be a large numbers of data points which cover large ranges of values. There are also various size constraints to be considered such as the size of the screen, symbols, fonts or other identifiers. One general information visualisation technique for management of this problem is to incorporate interactivity into the visualisation, first providing the user with an overview, then zooming and filtering, and obtaining the information details on demand. This principle is called the “Information Seeking Mantra” and was introduced by Shneiderman [1996]. Another technique is to display an overview of data but allow detailed information to be displayed simultaneously (also known as focus+context) such as with fish-eye.

There is a variety of textual information contained in many general datasets, and software datasets are no exception (for example class, method or package names, bug categories and rankings, and agile stories and tasks). Text is problematic for some multi-variate visualisation techniques (such as scatterplots or treemaps) due to space limitations. No one visualisation method or technique may provide a good balance for all considerations for a software engineering dataset and a combination of techniques may be necessary, using whichever method is most effective for a particular context.

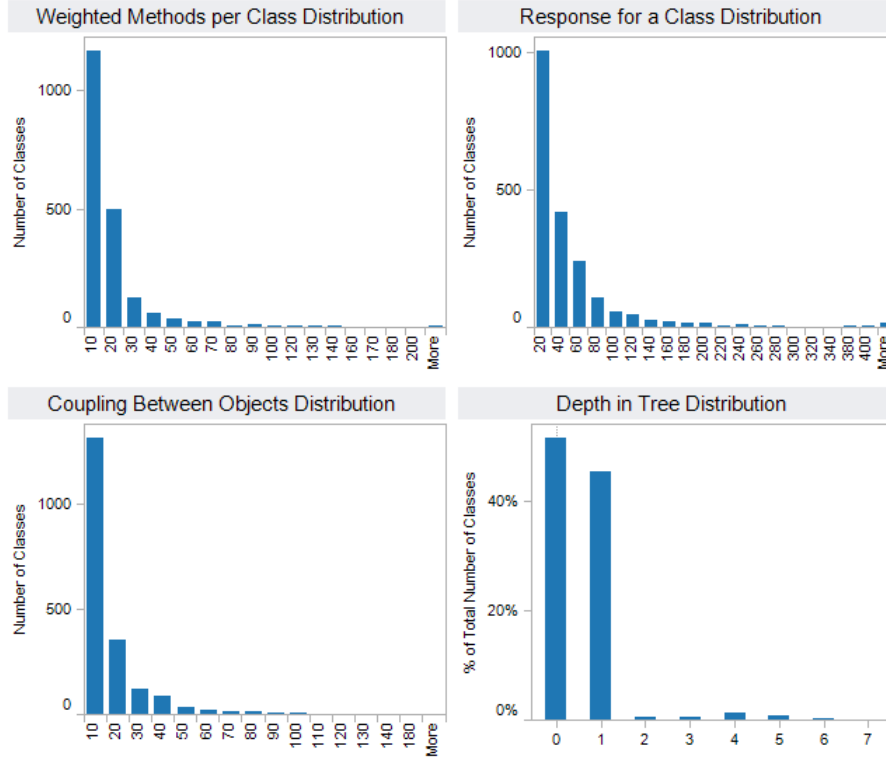


Figure 3.3: *Distribution of Chidamber and Kemerer metrics for open source project AspectJ (calculated by CKJM toolkit)*

3.3 Datasets for the tag cloud technique

One of the benefits of tag cloud visualisation is that data point textual identifiers (such as names) are an integral part of the graphic, meaning that users don't have to navigate into the visualisation to find important information. With this in mind, the sort of dataset that would be optimal to display in a tag cloud is one which includes modest amounts of textual information. Many datasets include this sort of information, in the form of names, labels or identifiers. In a general capacity, example datasets that might be used include names of people, brand names or marketing data, company financial information, stock market or foreign exchanges, country statistics, animal endangerment ratings, sporting events, file and folder names in a computer, and so on. Software engineering datasets also contain this sort of data such as class, method or package names, bug categories

and rankings, and agile stories and tasks. See Figures 3.4 and 3.5 for examples of a general and software related dataset.



Figure 3.4: *Popularity of baby names*

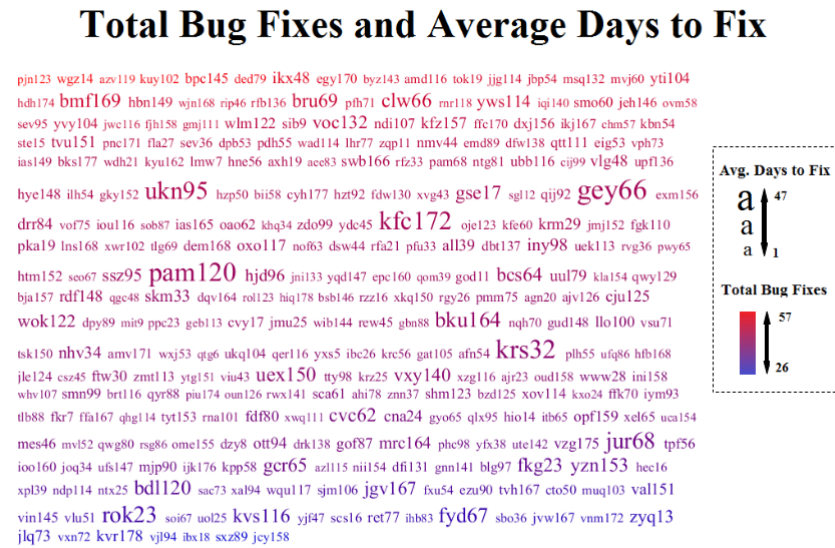


Figure 3.5: *Bug fixes for each user*

Much of the important textual data contained in these sorts of datasets is a phrase or collection of words rather than just one word. Also, many datasets containing some sort of label or identifier utilise many characters — such as a fully

qualified file or class name. It is important to note that multiple word phrases and lengthy label names can waste valuable real estate in the visualisation. There are two possible issues with this 1) it may have the effect of the data point appearing to have more importance than it actually does due to the increased prominence of the text, and 2) with multiple word tags it may be difficult to distinguish the boundaries between data points (this can be seen in Figure 3.6).

Figure 3.6: *Endangered species ranking for NZ birds*

- **Textual datasets:** *Optimal datasets contain textual identifiers such as class names or stories*
- **Filterable tag phrases:** *Multiple words waste valuable real estate in a visualisation, so should be filterable*
- **Establish tag boundaries:** *Boundaries between tags with multiple words should be clearly established*

3.4 Task types to enable data exploration in a tag cloud

It was suggested by [Rivadeneira et al. \[2007\]](#) in early empirical work, that the task set supported by tag clouds includes searching, browsing, impression forming and recognition/matching. Evidence exists that suggests tag clouds can provide improvements for summarising descriptive information (overviews) and also that they may be useful as a descriptive supplement for traditional search interfaces [[Kuo et al., 2007](#); [Sinclair and Cardew-Hall, 2008](#)]. However, research has compared tag clouds negatively to tables or lists for search and locate tasks [such as [Halvey and Keane, 2007](#); [Kuo et al., 2007](#); [Oosterman and Cockburn, 2010](#); [Rivadeneira et al., 2007](#)]. These experiments which report sub-optimal results for tag clouds required searching for or locating textual tag names and do not ask users to complete a visual search using additional features such as colour. (See Chapters 8 and 9 which investigate the possibility of the use of tag background colour or dual visual feature mapping improving user performance in a visual search.) Furthermore, we believe that more complex datasets (such as those with multiple data variables and intricate relationships between records) and other domain-specific factors, add more substance than the simplistic search and locate experiments might suggest.

We have identified an appropriate set of tasks shown in Table 3.2 on the following page which match the potential capabilities of tag clouds to tasks that are useful in software engineering (and also have a wider application in multi-variate data analysis). For categorisation purposes these can be further assigned to task types associated with data mining.

As an example of how tag clouds may be used to complete the associative task of identify correlations between variable, see Figure 3.7 on the next page which shows (artificially composited) composer popularity against the date of death. We can see that modern composers (who died later) are more popular than composers who died earlier such as those from the Classical or Baroque period. In Figure 3.5 on page 43 we can use the tag cloud font size to establish a general data distribution (summarising task). Most user names in the tag cloud are very small with only a few larger names, therefore we can hypothesise that

Table 3.2: *Tasks*

Task Description	Task Type
Identify similar characteristics of data	Clustering
Identify data distribution	Summarising
Identify data correlations	Associative
Detect outliers in a correlation	Summarising
Finding minimum/maximum values	Classifying
Comparison of data elements	Classifying

most users take a comparatively lower number of days to complete bug fixes.



Figure 3.7: *Popularity ranking and composer date of death*

3.5 Summary and discussion

The design considerations for an interactive tag cloud visualisation tool presented in this chapter have been developed through analysis of current tag cloud and information visualisation research, guidelines, and principles. We summarise these considerations in Table 3.3.

Design Considerations	Description
Textual datasets	Optimal datasets contain textual identifiers such as class names or stories.

Continued on next page

Table 3.3 – *Continued from previous page*

Design Considerations	Description
Filterable tag phrases	Multiple words waste valuable real estate in a visualisation, so should be filterable.
Establish tag boundaries	Boundaries between tags with multiple words should be clearly established.
Tag order	Visual property tag order is suitable for mapping to data fields.
Equal length tag identifiers	It should be possible to set the length of the tag identifiers to an equal length to minimise effects on user perception.
Font size	Visual property tag order is suitable for mapping to data fields.
Constrained font sizes	Font size should be constrained to greater than 9pt, and a suitable maximum font size according to canvas and screen boundaries.
Comparable tags	Tags should be able to be compared by moving closer together to assist quantitative comparisons.
Font family	Visual property tag order is not suitable for mapping to data fields.
Familiar fonts	For reading accuracy, fonts should be familiar such as Times New Roman, Arial or Helvetica.
Colour hue	Visual property colour hue is suitable for mapping to data fields.
Colour transparency	Visual property colour transparency is suitable for mapping to data fields.
Colour selection	Colour codes should be taken from Ware’s colour code recommendations (red, green, yellow, black, blue, white, pink, cyan, grey, orange, brown, and purple).
Colour background	Visual property colour background is suitable for mapping to data fields.

Continued on next page

Table 3.3 – *Continued from previous page*

Design Considerations	Description
Colour contrasts	Strongly contrasting colour schemes should be selected such as red, blue and green on a white background.
Font styles	Visual property font styles are not considered suitable for mapping to data fields.

In §3.4, we presented a set of task types that we believe should be supported in a tag cloud visualisation tool in order to enable exploration of a software engineering dataset. These task types were generated from analysing software visualisation challenges and tag cloud capabilities. The task types and design considerations were used to inform the design of our interactive tag cloud visualisation system ‘Taggle’, presented in Chapter 4.

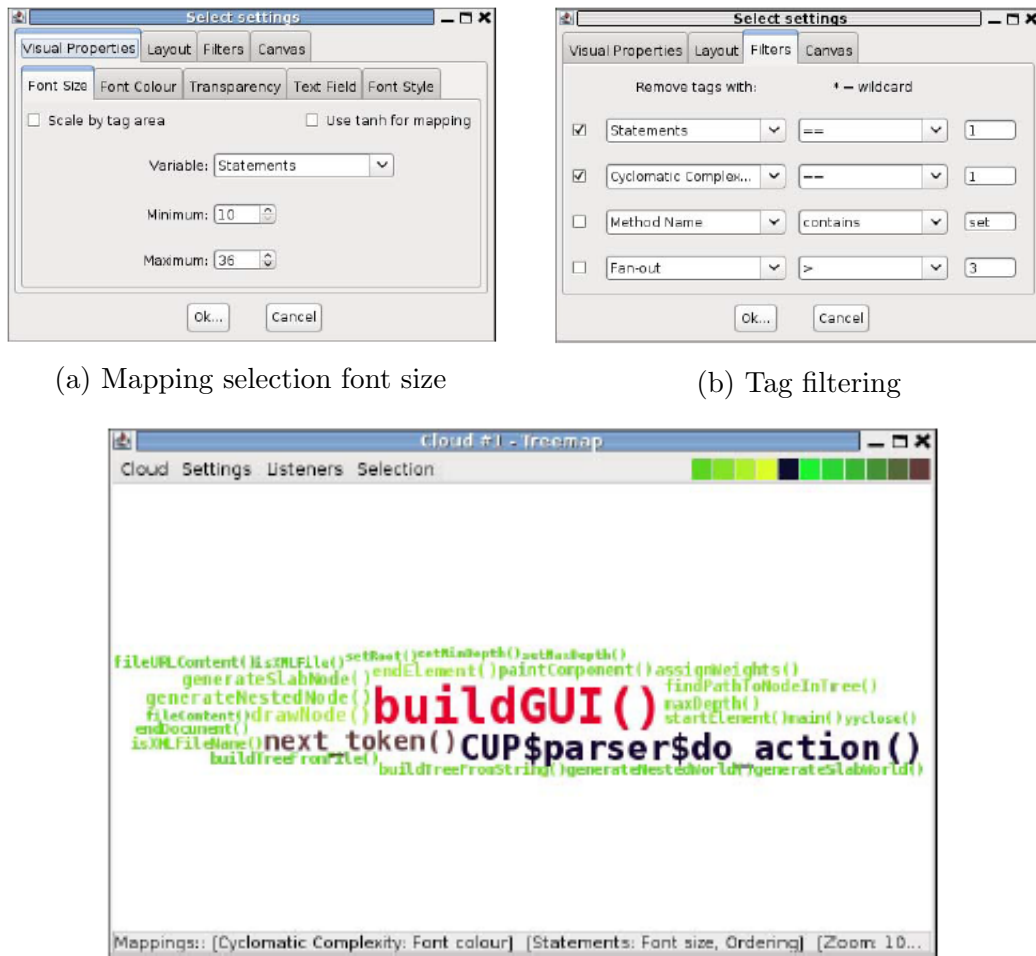
4

Taggle: A Tag Cloud Visualisation Tool

To more effectively support exploration of unknown datasets in software engineering, we designed and implemented a tag cloud visualisation tool, Taggle, targeted at exploration of software quality assurance metric data. Guided by the design considerations of Chapter 3, the system uses the visual properties of tag clouds to represent data fields, and provides rich interactive features such as dynamic and static filtering to support knowledge discovery. This chapter describes the details of Taggle’s implementation; in §4.1 we first discuss the original inherited prototype and changes made in the course of this research. In §4.2 we then review the Taggle data model and describe how data is transformed to fit into this model. Visual encoding used to render the transformed data for display and mapping selection is discussed in the §4.3. Finally, in §4.4 we present options to complete a set of software engineering tasks using Taggle.

4.1 Original prototype

An interactive tag cloud visualisation tool implemented using the Java 2D API was inherited from previous University of Canterbury research. Some aspects of the original prototype are shown in Figure 4.1 and descriptions can be found in technical documents: Churcher et al. [2011]; Deaker and Churcher [2011]; Deaker et al. [2011].



(a) Mapping selection font size

(b) Tag filtering

(c) Canvas with tag cloud and colour chip legend

Figure 4.1: *Original Taggle prototype*

In this research, I have extended and enhanced this prototype in order to create a stable system which could satisfactorily explore software quality assurance

data such as metrics and code smells using tag cloud visualisation. Additionally, the possibility of using other software artefacts such as process management data and general multi-variate data, has been left open.

In the course of this research, changes were made to the original prototype which included:

- implementation of changes resulting from software engineering challenges and tag cloud visual variable analysis (Chapter 3)
- improvements generated from heuristic evaluation (Chapter 7)
- implementation of an XML data conversion utility
- usability improvements and bug fixes

The software pictured and described in the rest of this chapter refer to the final prototype, after all alterations and improvements have been made.

4.2 Data model and transformation

Source data for Taggle may be generated externally and provided in an XML format conforming to a DTD. There are a variety of tools/methods which can produce software metric data from static analysis of source code (such as that proposed by [Irwin and Churcher \[2003\]](#), and see Appendix A for a list of available plugins, software analysis platforms and frameworks/tools that can generate metric data). Most of these tools can output data into a CSV or XML format (§A.3). I produced a tool to convert CSV formatted data to the Taggle XML format. This has also proved useful for conversion of datasets that may easily be obtained from external sources such as www.findthedata.com. Tools which produce XML formatted output only may also be transformed (using XSLT for example).

The Taggle XML format specifies such things as measurement scales (nominal, ordinal and ratio) and relationships. Datasets consist of records (data points), each containing a number of fields. Visual mappings to properties and constraints for each data field can be specified via the GUI. Once the constraints of the visual

property are set, a relative weighting for each tag is calculated according to the rank of the measurement. The calculation of rank is dependent on the measurement scale type. The weightings are used to calculate the value of the visual property for an individual tag, between the user defined constraints for a visual property (for example minimum and maximum font sizes). A full description of the Taggle XML format and weighting calculations can be found in [Deaker and Churcher \[2011\]](#).

As an example, consider the tag cloud in [Figure 4.2 on the following page](#): the XML source file contained quality metrics from a subproject of open source project ActiveMQ, the metrics were of the Chidamber and Kemerer suite generated by the CKJM toolkit. These were originally produced in space-separated text format, converted to CSV by OpenOffice, and then converted to the Taggle XML format by our conversion tool. In the GUI, the nominal field “Class” was mapped to the tag text (Arial font), the ratio field “LCOM” was mapped to the tag ordering property and the font size property (spanning from 20pt to 35pt), and the ratio field “CBO” was mapped to a background colour range from black to red. The tags have been positioned according to a simple tag cloud layout algorithm “typewriter”, where the tags are mapped sequentially left to right, top to bottom (ways to improve the overall presentation are discussed later in the chapter). Default values for the visual mappings and other settings for Taggle can be controlled using an external XML file.

4.3 Visual encoding and mapping selection

For brevity, only visual encoding and mapping features utilised in this research are discussed in this section. Information regarding other mapping properties and tag cloud layout algorithms can be found in the following technical documents: [Deaker and Churcher \[2011\]](#); [Deaker et al. \[2011\]](#) (note that these documents describe the basic procedure which hasn’t changed much, whereas the graphical interface, numbers of options, and features have changed significantly through the course of this research).

The mapping selection interface can be seen in [Figure 4.3 on page 54](#). In the analysis of visual properties available in tag clouds ([Chapter 3](#)), order, size,

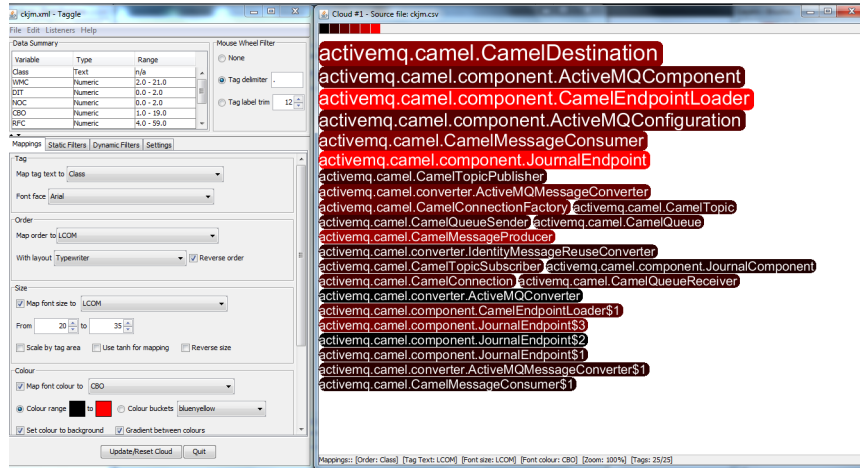


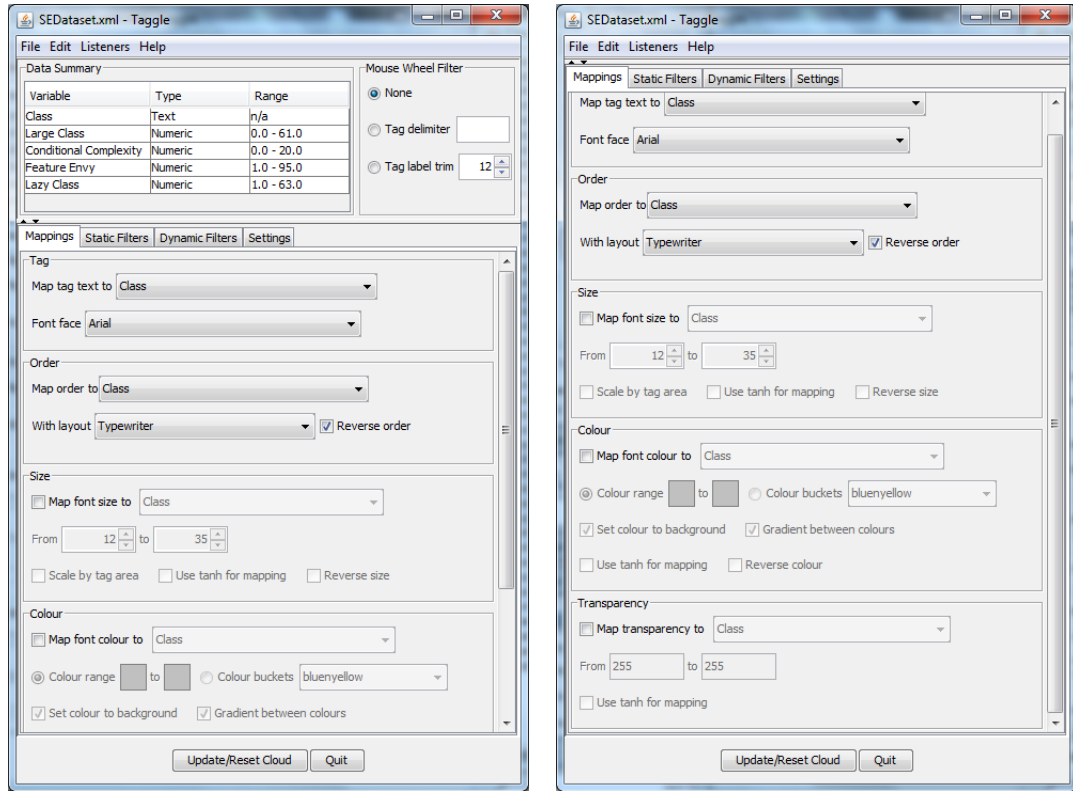
Figure 4.2: A tag cloud generated by Taggle depicting a subproject of ActiveMQ with metrics derived from CKJM. Visual mapping interface is on the left.

colour and transparency were deemed the most appropriate visual variables for data mapping purposes. The order in which they are displayed on the central mapping pane is in order of perceived usefulness and importance (with tag order being the most useful and transparency the least useful). By default, only the minimum mappings essential for display (tag and tag order) are turned on.

The user is supported in the choice of appropriate mappings for the tag cloud by a data summary screen shown in a resizeable pane at the top of the GUI — for an example see Figure 4.4 on the following page. Each field in the data file is shown along with data type (text, numeric or categorical) and the range of values in the category. The data summary panel was included by request from a heuristic evaluation (see Chapter 7).

4.3.1 Categorical and continuous data

Data types displayed in the data summary screen are determined programmatically. Fields are labelled categorical when the value ranges correspond to a certain maximum distinct number of values (the actual number is controlled via a setting). It is limited to a particular number of distinct values to match the number of colour chips displayed in the legend — categorical variables are used predominantly in display colour mappings (§4.3.5). Providing a method for distinguishing



(a) Data summary panel toggled on

(b) Data summary panel toggled off

Figure 4.3: Mapping selection interface: the data summary panel can be toggled by the user as desired to save valuable screen space

Data Summary		
Variable	Type	Range
Runner ID	Numeric	1.0 - 100.0
Surname	Text	n/a
Club	Category	Aldershot F, Ald...
Club ID	Numeric	1.0 - 5.0
Score	Numeric	2.0 - 16.0
Rand1	Numeric	1.0 - 101.0

Figure 4.4: Closeup of an example data summary panel

these two types of data was included by request from the heuristic evaluation, where it was determined to be useful to display categorical data using a unique

colour for each category.

4.3.2 Tag

Visual variable: tag length The user is free to select any field to map to the tag text, including numeric types. The tag lengths for each tag will differ depending on the number of characters in the tag text. To minimise the possible effects on perception long textual identifiers may have, see §4.4.1 for a description of the application of filters.

Visual variable: font family The font face dropdown box is pre-populated with fonts that are installed on the machine running Taggle. Because reading ease is improved when presented with familiar fonts [pg 106, chap 11:7 [Health and Services, 2006](#)], Arial, Helvetica and Times New Roman (or the equivalents depending on the operating system and installed word processing software) are presented for selection at the top of the dropdown box, separated by a divider (see Figure 4.5).

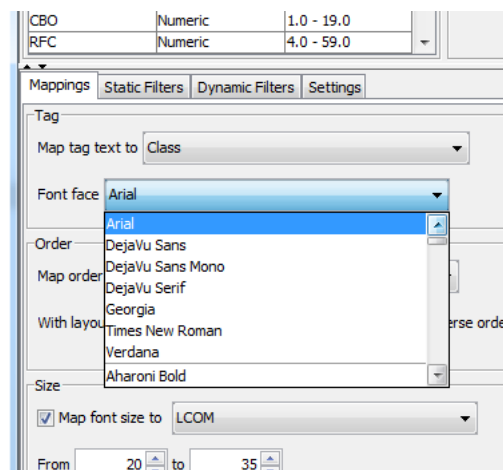


Figure 4.5: *Prompting selection of familiar fonts.*

4.3.3 Order

Visual variable: order If the order visual property is mapped to a field containing textual data, ordering of the tags is determined alphabetically, otherwise

tags are ranked by their numerical value. The selected ordering can be reversed with the “reverse order” checkbox and updating the tag cloud. The user can select an appropriate layout from typewriter, spiral or force directed (defaulting to the simplest layout, typewriter).

4.3.4 Size

Visual variable: font size Research has indicated slower reading performance for smaller font size [pg 107, chap 11:8 [Health and Services, 2006](#)], so there are constraints on the minimum font size to be set to no less than 9pt. The maximum font size is constrained to 250pt: by default this is set to a much smaller 35pt to allow for a smaller screen and canvas size (canvas size can be adjusted from the settings tab). The data mapped to the selected font sizes can be reversed with the “reverse size” checkbox.

4.3.5 Colour and transparency

Visual variable: background colour The heuristic evaluation and controlled experiments conducted in this research determined that the use of background colour was useful for visual search with colour mappings, as well as dataset features such as multiple word tags. Background colour has also been found to support more accurate estimation of relevant tags [[Waldner et al., 2013](#)]. Therefore tag display via background colour is the default setting in Taggle.

The user may select from a choice of colour range (between user defined colours) or colour buckets (a set palette of colours) — see [Figure 4.6 on the following page](#). When the user chooses a data field to map to colour, colour range or colour bucket is automatically selected depending on whether the data field is categorical data. Categorical data is best displayed through a unique set of colours preselected from a colour palette (the colour palettes can be user defined through external string property files, although default palettes are provided.) For example, in [Figure 4.7 on the next page](#) a dataset about agile projects is displayed in a tag cloud. With the aim of finding what actors are related to the agile stories with the highest time estimates, order/size are mapped to estimate and colour is mapped to actor. The user can determine stories with the greatest

estimated workload corresponding to tags coloured dark grey and brown, and find details via a colour legend mouse hover or tag label mouse hover (actors ‘Preferred customer’ or ‘Bookstore customer’).

Continuous data is best displayed through a colour range. The user has the option of choosing a colour gradient between two selected colours, or colours which appear on the colour wheel between two selected colours (see Figure 4.8 on the following page. Colour models are complicated and won’t be discussed in full detail). Like other mappings, the user also has the option to reverse the selected mappings so high values can be shown with the ‘from’ colour and low values can be shown with the ‘to’ colour.

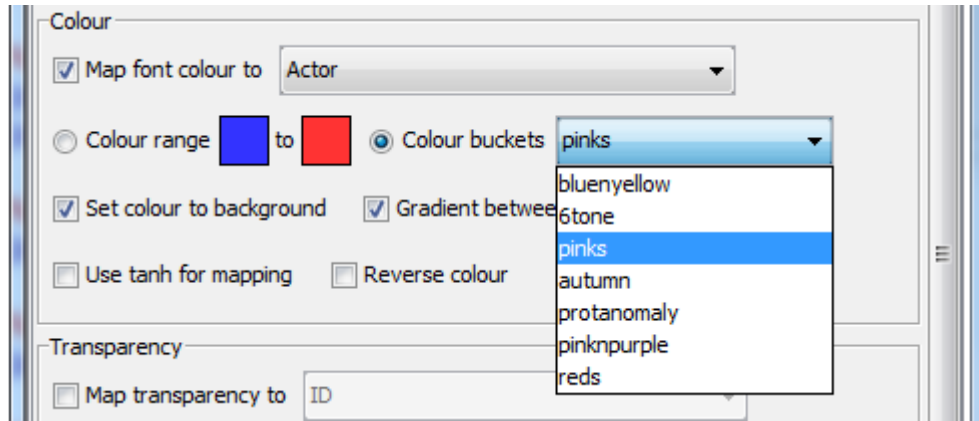
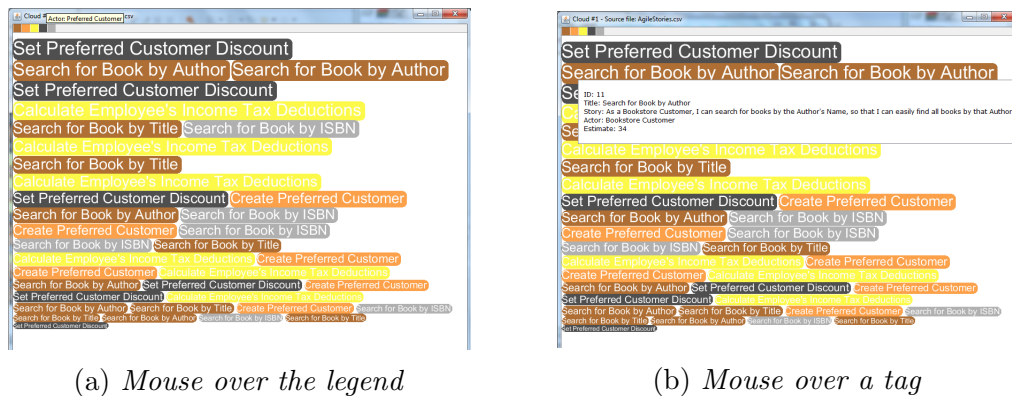


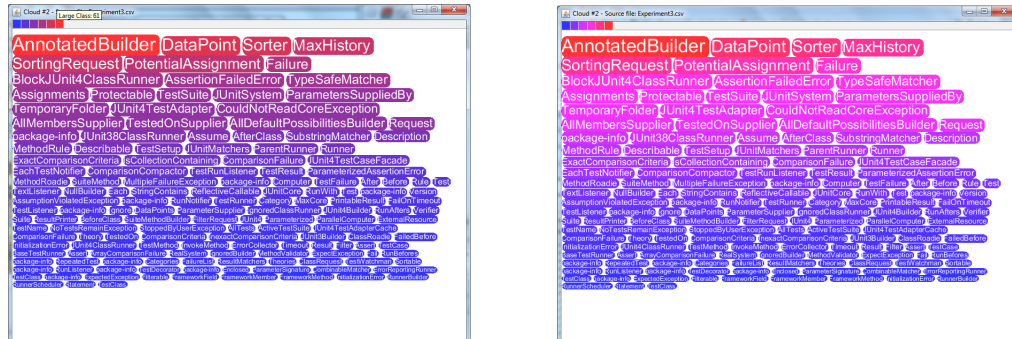
Figure 4.6: *Colour mapping interface*



(a) *Mouse over the legend*

(b) *Mouse over a tag*

Figure 4.7: *Displaying categorical data with colour buckets*



(a) *Colour gradient between red and blue* (b) *Colour wheel between red and blue*



(c) *A colour wheel*

Figure 4.8: *Displaying continuous data with colour ranges*

Visual variable: font colour The user also has the choice of displaying a more conventional looking tag cloud, with colour mapped to the font rather than the background (see Figure 4.9 on the next page).

Visual variable: transparency The user may map data fields to a visual property which is familiarly used in tag clouds online — transparency — see Figure 4.10 on the following page. This can be applied with or without colour.

4.4 Software engineering tasks

Taggle includes a number of interactive features to enable rich data exploration. Only the features which were utilised in this research and pertain directly to

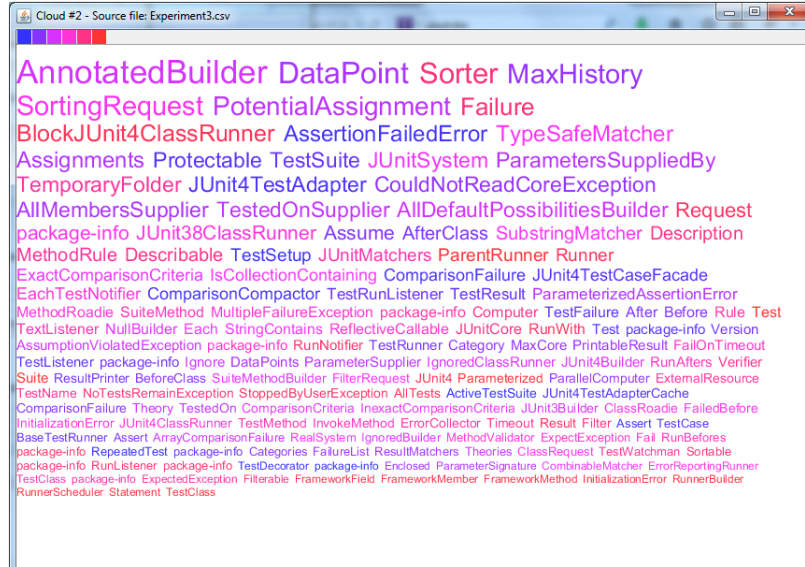
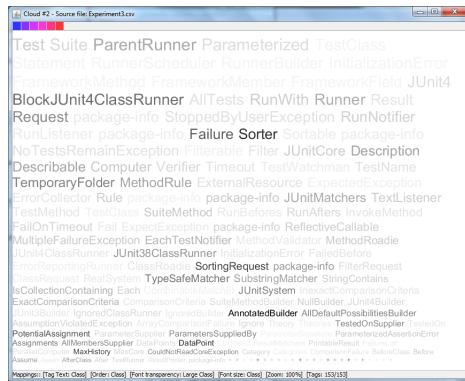
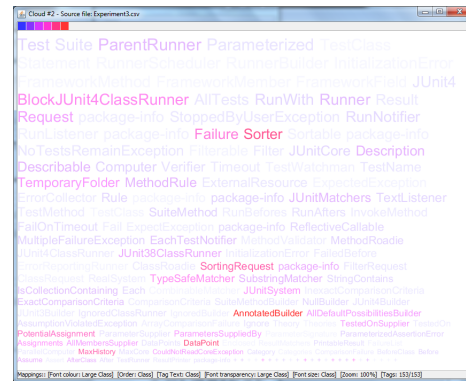


Figure 4.9: *Colour mapped to the font*



(a) *Without colour applied*



(b) *With colour applied*

Figure 4.10: *Transparency mapping*

completing the software engineering tasks outlined in §3.4 are described in this section. For further information describing features such as relationship highlighting, cloud listeners, sub-clouds and linking, see technical documents: [Deaker and Churcher \[2011\]](#); [Deaker et al. \[2011\]](#).

4.4.1 Filtering textual data

Textual data is an issue with some multi-variate data visualisation techniques such as treemaps or scatterplots because of issues with screen real estate. On the other hand, a tag cloud is designed to include the (often important and identifying) textual data as a key part of the graphic. Software textual data is often in the form of long identifiers such as class or agile story names. Taggle’s mouse wheel filtering (used for textual data displayed in the tag, interface shown in Figure 4.11) can be used to dynamically select the portions of the text identifier that the user deems most important and/or maximise the available visualisation space. When the ‘Tag delimiter’ option is selected, the user may filter the text by a particular character by scrolling the mouse wheel (scrolling up filters off leading text while scrolling down filters off trailing text). In Figure 4.12 on the following page you can see an example of typical software engineering data displaying Java package and class names as the textual identifier in the text with a) showing the original tag text and b) text filtered to two packages above the class name.

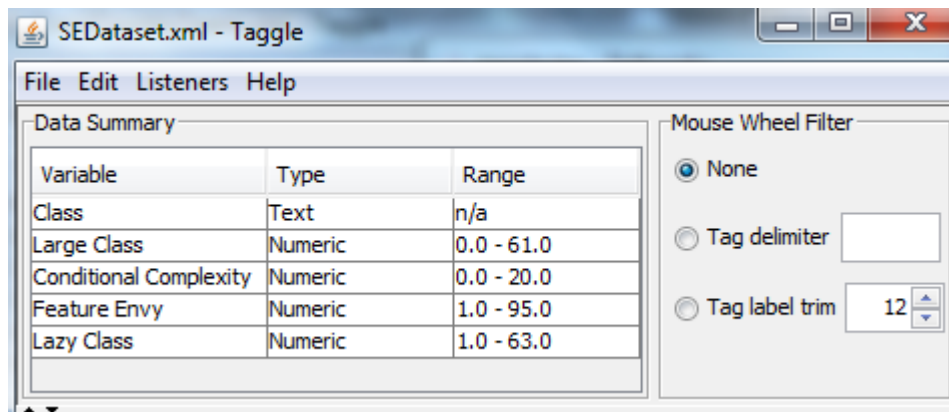


Figure 4.11: *Mouse filtering interface*

The user also has the option of filtering the text to a fixed number of characters from either the leading or trailing text (Figure 4.13 on page 62). This has two purposes, to maximise available real estate in the canvas, and to minimise any effect a greater number of characters in a tag may have on user perception. This filter can be used in combination with removing the font size mapping selection, which further minimises bias in eye attention.


```

org.junit.Test org.junit.runners.Suite org.junit.runners.ParentRunner
org.junit.runners.Parameterized org.junit.runners.model.TestClass
org.junit.runners.model.Statement org.junit.runners.model.RunnerScheduler
org.junit.runners.model.RunnerBuilder org.junit.runners.model.InitializationError
org.junit.runners.model.FrameworkMethod org.junit.runners.model.FrameworkMember
org.junit.runners.model.FrameworkField org.junit.runners.JUnit4
org.junit.runners.BlockJUnit4ClassRunner org.junit.runners.AllTests org.junit.runner.RunWith
org.junit.runner.Runner org.junit.runner.Result org.junit.runner.Request
org.junit.runner.package-info org.junit.runner.notification.StoppedByUserException
org.junit.runner.notification.RunNotifier org.junit.runner.notification.RunListener
org.junit.runner.notification.package-info org.junit.runner.notification.Failure
org.junit.runner.manipulation.Sorter org.junit.runner.manipulation.Sortable
org.junit.runner.manipulation.package-info org.junit.runner.manipulation.NoTestsRemainException
org.junit.runner.manipulation.Filterable org.junit.runner.manipulation.Filter org.junit.runner.JUnitCore
org.junit.runner.Description org.junit.runner.Describable org.junit.runner.Computer org.junit.rules.Verifier
org.junit.rules.Timeout org.junit.rules.TestWatchman org.junit.rules.TestName org.junit.rules.TemporaryFolder
org.junit.rules.ErrorCollector org.junit.rules.ExternalResource org.junit.rules.ExpectedException
org.junit.rules.Rule org.junit.rules.package-info org.junit.matchers.package-info
org.junit.matchers.JUnitMatchers org.junit.internal.TextListener org.junit.internal.runners.TestMethod
org.junit.internal.runners.TestClass org.junit.internal.runners.SuiteMethod org.junit.internal.runners.statements.RunBefore
org.junit.internal.runners.statements.RunAfter org.junit.internal.runners.statements.InvokeMethod
org.junit.internal.runners.statements.FailOnTimeout org.junit.internal.runners.statements.Fail
org.junit.internal.runners.model.MultipleFailureException org.junit.internal.runners.model.EachTestNotifier
org.junit.internal.runners.MethodValidator org.junit.internal.runners.MethodRoadie org.junit.internal.runners.JUnit4ClassRunner
org.junit.internal.runners.JUnit38ClassRunner org.junit.internal.runners.InitializationError org.junit.internal.runners.FailedBefore
org.junit.internal.runners.ErrorReportingRunner org.junit.internal.runners.ClassRoadie

```

(a) Original tag text

```

unit.Test unit.runners.Suite unit.runners.ParentRunner
unit.runners.Parameterized junit.runners.model.TestClass
unit.runners.model.Statement unit.runners.model.RunnerScheduler
unit.runners.model.RunnerBuilder junit.runners.model.InitializationError
unit.runners.model.FrameworkMethod junit.runners.model.FrameworkMember
unit.runners.model.FrameworkField junit.runners.JUnit4
unit.runners.BlockJUnit4ClassRunner junit.runners.AllTests junit.runner.RunWith
unit.runner.Runner junit.runner.Result junit.runner.Request junit.runner.package-info
unit.runner.notification.StoppedByUserException junit.runner.notification.RunNotifier
unit.runner.notification.RunListener junit.runner.notification.package-info
unit.runner.notification.Failure junit.runner.manipulation.Sorter junit.runner.manipulation.Sortable
unit.runner.manipulation.package-info junit.runner.manipulation.NoTestsRemainException
unit.runner.manipulation.Filterable unit.runner.manipulation.Filter junit.runner.JUnitCore
unit.runner.Description unit.runner.Describable unit.runner.Computer unit.rules.Verifier
unit.rules.Timeout unit.rules.TestWatchman unit.rules.TestName unit.rules.TemporaryFolder
unit.rules.MethodRule junit.rules.ExternalResource junit.rules.ExpectedException junit.rules.ErrorCollector
unit.Rule junit.package-info junit.matchers.package-info junit.matchers.JUnitMatchers junit.internal.TextListener
unit.internal.runners.TestMethod junit.internal.runners.TestClass junit.internal.runners.SuiteMethod
unit.internal.runners.statements.RunBefore junit.internal.runners.statements.RunAfter
unit.internal.runners.statements.InvokeMethod junit.internal.runners.statements.FailOnTimeout
unit.internal.runners.statements.Fail junit.internal.runners.statements.ExpectException junit.internal.runners.package-info
unit.internal.runners.model.ReflectiveCallable junit.internal.runners.model.MultipleFailureException
unit.internal.runners.model.EachTestNotifier junit.internal.runners.MethodValidator junit.internal.runners.MethodRoadie
unit.internal.runners.JUnit4ClassRunner junit.internal.runners.JUnit38ClassRunner junit.internal.runners.InitializationError
unit.internal.runners.FailedBefore junit.internal.runners.ErrorReportingRunner junit.internal.runners.ClassRoadie
unit.internal.runners.RealSystem junit.internal.runners.TypeSafeMatcher junit.internal.runners.SubstringMatcher

```

(b) Filtered on a '.' delimiter

Figure 4.12: Filtering on a character delimiter



Figure 4.13: *Filtering on a fixed number of characters from the leading text*

4.4.2 Dealing with large scale data

One feature of software engineering datasets is that there is often a large number of data points to contend with, and any potential visualisation tool must provide a mechanism to deal with this. Using the “Information Seeking Mantra” [Shneiderman, 1996], we first provide the user with an overview of the dataset (see Figure 4.14 on the following page), and allow additional details to be accessed on demand via a mouse hover. As seen in Figure 4.12 on the previous page, tag labels too long to be displayed on the canvas are minimised into symbols. This has the advantage that they don’t disappear from view together, and hovering over the iconified form reveals their details. As the user interacts with the data using filtering and dynamic queries, these symbols are transformed into textual labels as canvas real estate becomes available.

Filtering can be applied to the dataset (Figure 4.15 on the following page) to narrow the view as desired, or dynamic querying (Figure 4.16 on page 64) where the user is able to see the tags filtered from display in real time using a range slider.

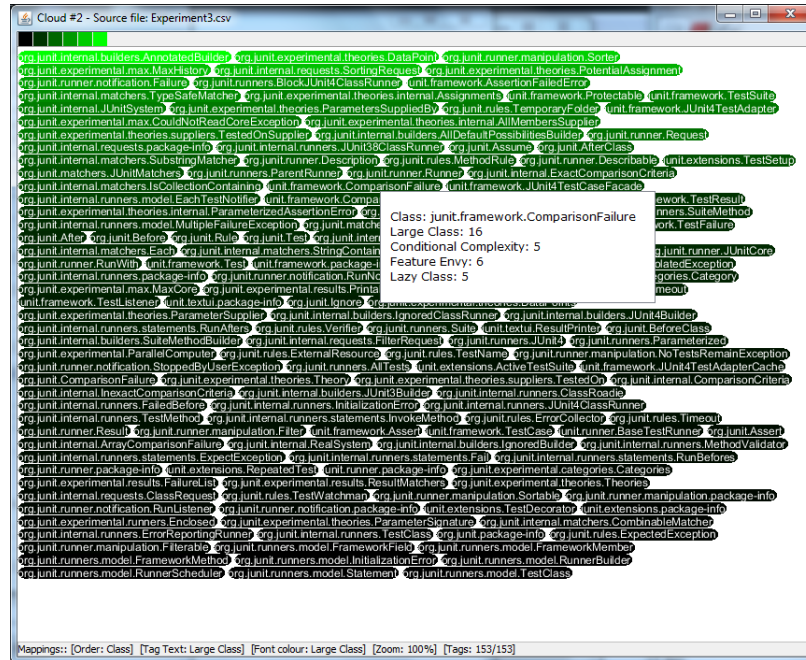


Figure 4.14: Overview first, then details-on-demand

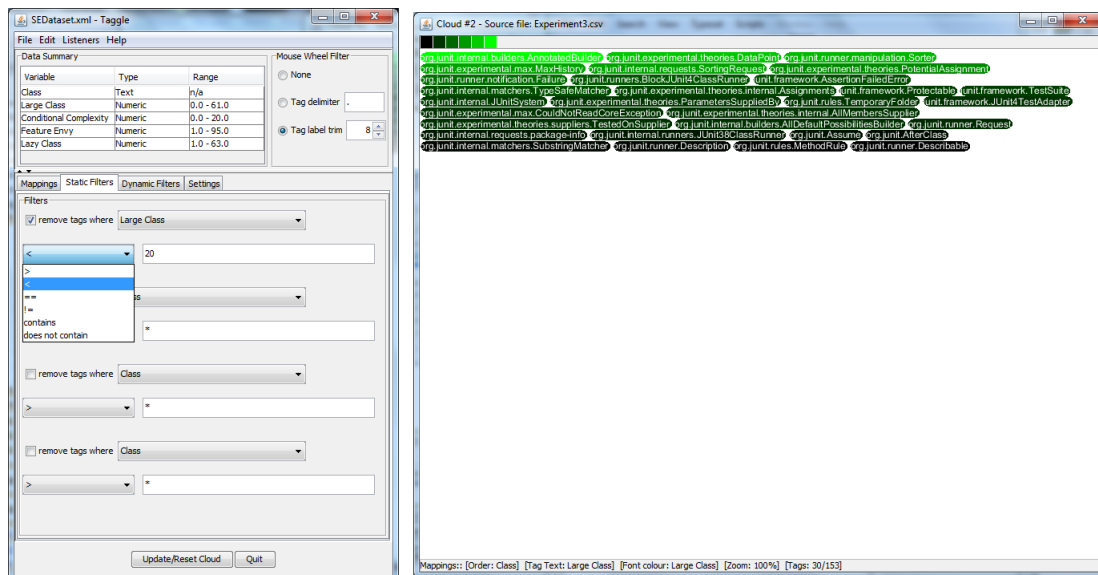


Figure 4.15: Applying filtering to the data

It is also possible to display an overview of data while allowing detailed information to be shown simultaneously with a zoom function (Figure 4.17 on page 65). A particular concern using the tag cloud technique is lack of space

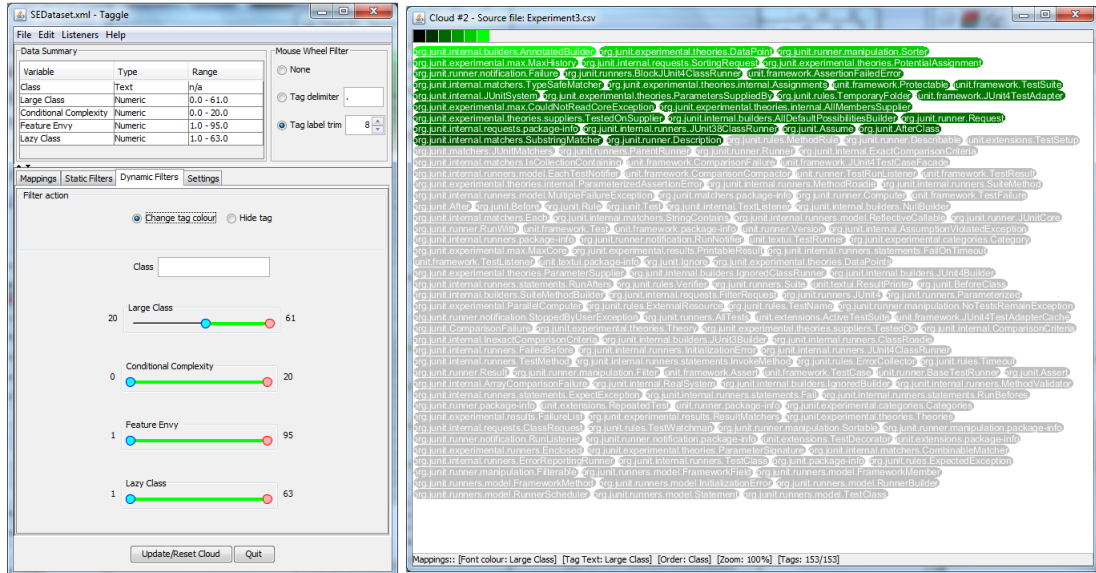


Figure 4.16: *Dynamic querying*

due to long label lengths, or many labels with a large dataset. In this case the user may apply dynamic mouse wheel filtering and label trimming as detailed in §4.4.1.

4.4.3 Identify similar characteristics of data

Similar data characteristics can be identified using filtering and details-on-demand. In Figure 4.18 on page 66, a user inspects the classes with a low level of the ‘lazy class’ metric within a software dataset. Using either dynamic queries or details-on-demand on the filtered dataset it is possible to establish that classes with a low level of ‘lazy class’ also have a low level of metric ‘feature envy’ (refer to Chapter 10 for further details regarding this example dataset). Following this kind of filtered inspection, a user might make use of a visual property mapping such as colour to identify a data correlation between the two variables (see §4.4.5).

4.4.4 Identifying data distribution

The distributions of variables in software engineering metric datasets are often heavily skewed (see Figure 3.3 on page 42 for some graphical examples of this).

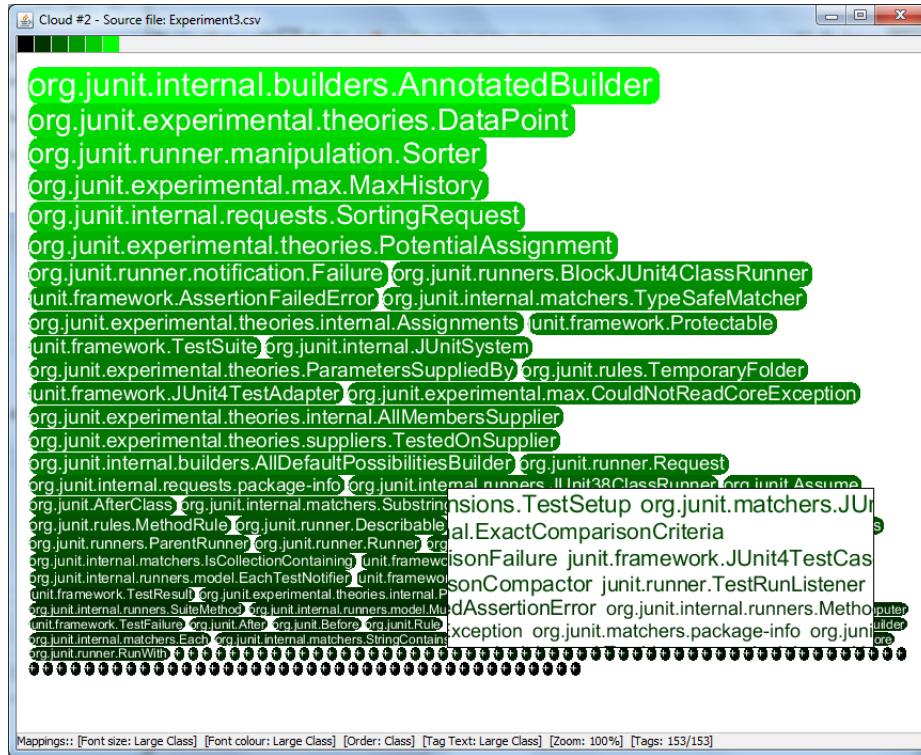


Figure 4.17: *Zoom in on a small area*

In Taggle, colour and order are mapping choices which can be applied to the data in order to highlight data distribution. In Figure 4.19 on page 67, the metric ‘large class’ is being visualised from a software project. To avoid bias in user perception, font size is unmapped and the tag labels have been filtered to a fixed length. The ordering of the tags is arranged in order of the metric in question, from largest value to smallest. A colour gradient between green and black is also mapped to ‘large class’, where black identifies tags which have the lowest levels of ‘large class’, and green shows tags with the highest levels. The distribution of colour across the visualisation shows the user that values of ‘large class’ are skewed towards the lower end as expected. As a user of Taggle, the classes that would now become of interest as possible candidates for refactoring would be the few brightest green tags to the top left.

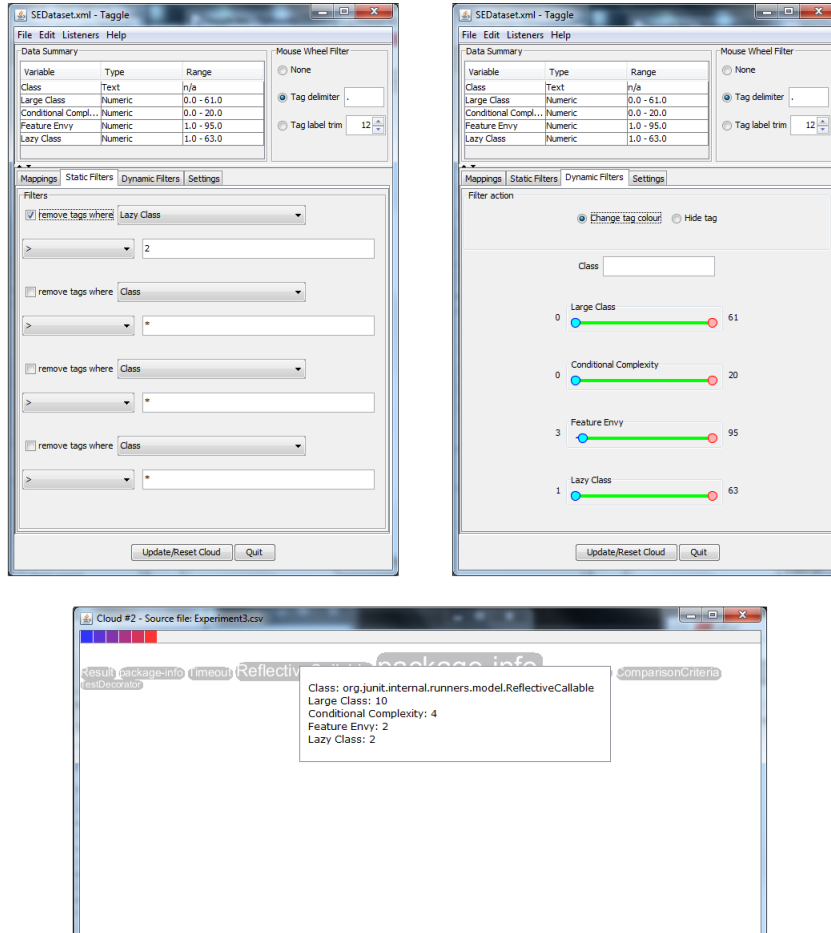


Figure 4.18: *Using filtering to identify similar data characteristics*

4.4.5 Identify data correlations and outliers

Software metric data distributions are typically heavily skewed and may contain outliers. In statistics, outliers are often discarded or ignored during analysis: in software engineering these outliers are often points of interest. In Figure 4.20 on page 68, colour and order are used to investigate a possible correlation between data variables, metrics 'lazy class' and 'feature envy'. Size and order are dually mapped to 'feature envy' while a colour range (blue to red) is mapped to 'lazy class'. We see that as the font size becomes smaller, the colour mapping changes gradually from red to blue in a progressive fashion. This is indicative of a correlation between the two variables. There are three data points which

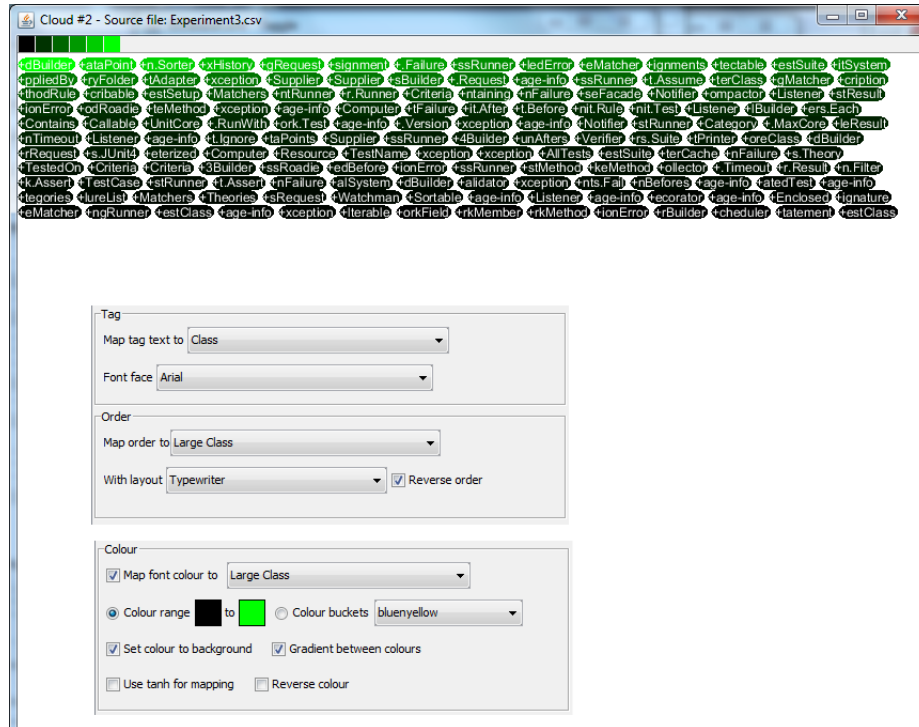


Figure 4.19: Using colour to identify data distribution of the ‘large class’ metric

have colouring that stands out amongst the rest (classes ‘Enclosed’, ‘Theory’ and ‘JUnit4TestAdapter’), these points may appear as outliers in the correlation between the data variables. In Figure 4.21 on the next page, colour is scattered without pattern across the order and font size, no correlation can be seen between variables ‘conditional complexity’ and ‘feature envy’.

4.4.6 Finding minimum/maximum values

In software metric analysis, finding classes which hold the minimum or maximum values of a metric is useful for identifying classes that are candidates for refactoring. The order mapping, or a dual mapping of order and font size, are useful for identifying classes in the lower and upper boundaries. In Figure 4.22 on page 69, ‘AnnotatedBuilder’ and ‘DataPoint’ classes are easily identified as having the highest values of the ‘conditional complexity’ metric when dually mapped to order and font size.

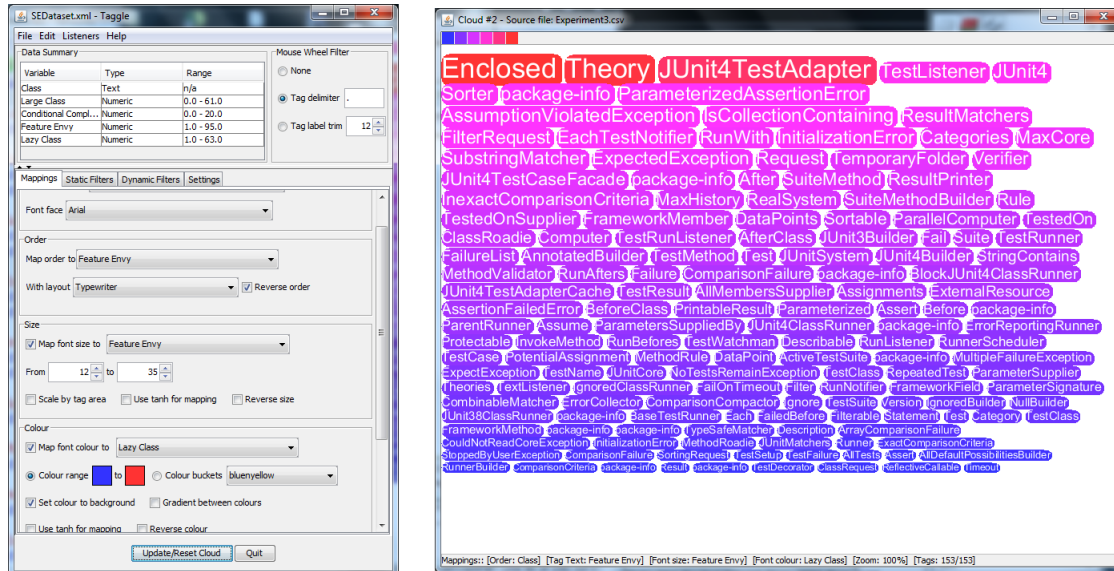


Figure 4.20: Using colour to identify a data correlation and outliers

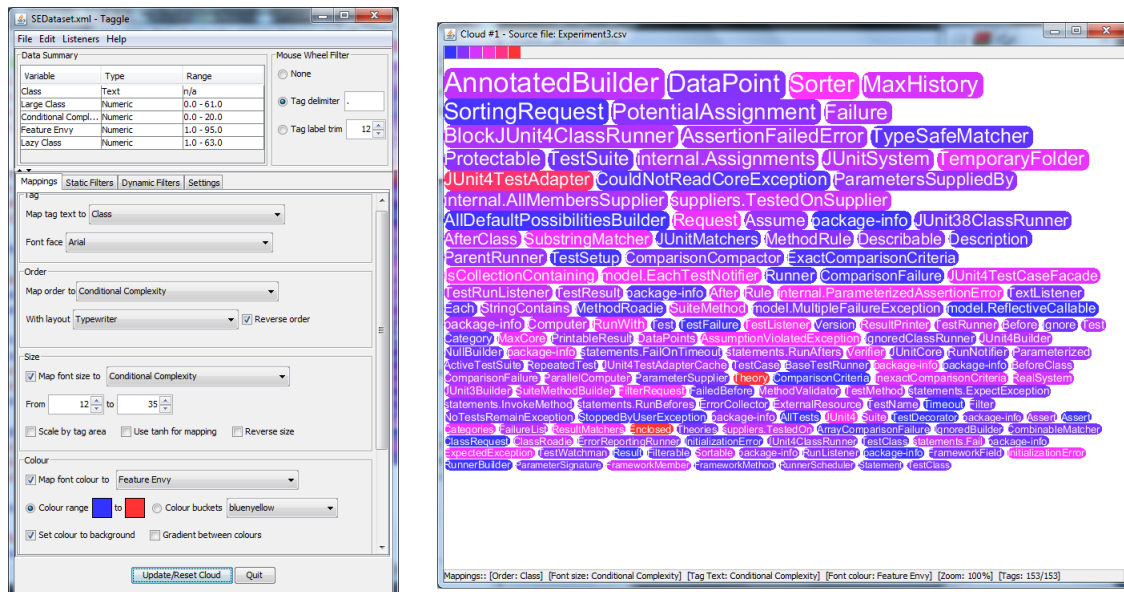


Figure 4.21: No data correlation between variables

4.4.7 Comparison of data elements

Comparison of data points is another key task useful in any software engineering or multi-variate dataset. In Figure 4.22 on the next page font size and order are dually mapped to field 'conditional complexity'. It is obvious for example, that

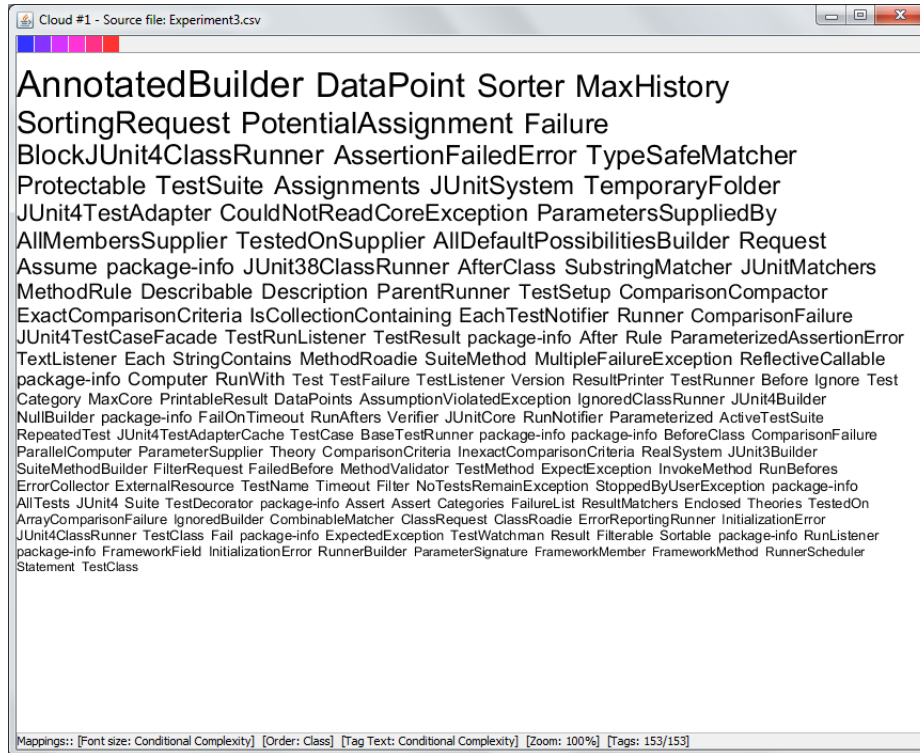


Figure 4.22: *Finding classes with the highest value of a metric using dual order and font size mappings*

tag ‘JUnitSystem’ has a higher conditional complexity than tag ‘JUnitMatchers’. When font size and order are not mapped to the same data field it may not be so obvious. Using drag and drop a user may pull out the relevant tags and put them next to one another for easier comparison — see Figure 4.23 on the following page. Additionally, a user may hover over the tags and view the class metric information in the pop-up details-on-demand for comparison.

4.5 Summary and discussion

In this chapter, we introduced the tag cloud visualisation tool Taggle. This tool was inherited from previous research and extensively modified to create a stable system able to explore software quality metrics and other more general multi-variate data. Most notably, a variety of implementation changes were made as a result of considering design aspects from software engineering and tag cloud

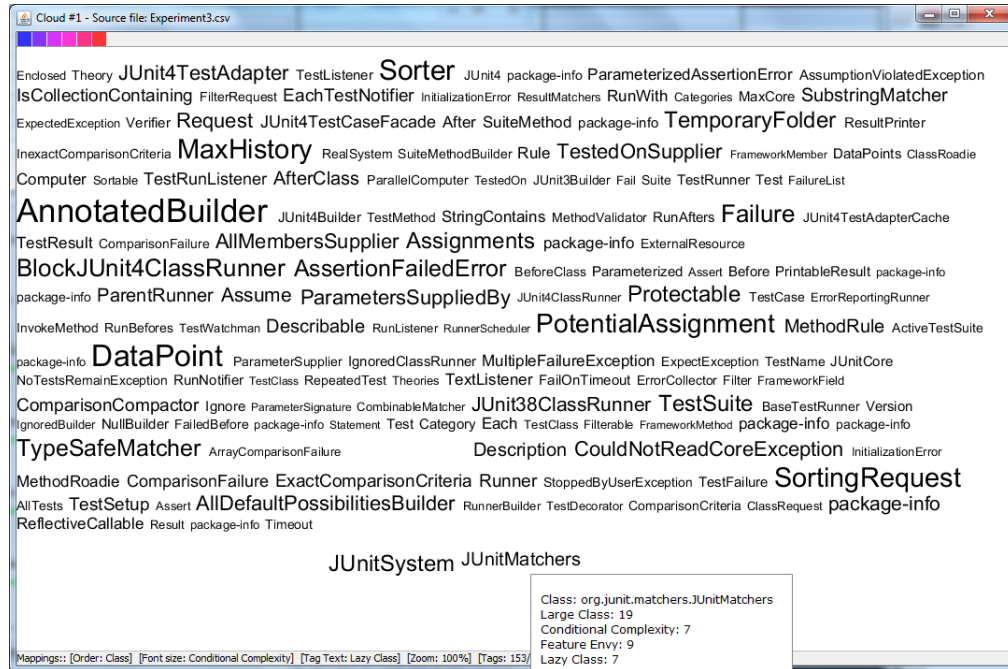


Figure 4.23: *Comparison of ‘JUnitSystem’ and ‘JUnitMatchers’*

visual variable analysis (Chapter 3) as well as improvements generated from a heuristic evaluation by domain experts (Chapter 7).

Data can be gathered from a variety of sources and then converted (through XML transform or conversion utility) into a special Taggle XML format. This XML file is then input into Taggle through the user interface, where the user can view generated tag clouds and customise the visual property mappings as they explore the dataset. Interface and visual encoding design choices such as the visual properties included, constraints and options for each property, and filtering selections were made with respect to the software engineering and design considerations listed in previous chapters. Some choices, such as the data summary screen and categorical data options, were incorporated by request during the heuristic evaluation.

In Chapter 3, a list of tasks was presented that represented meaningful ways users could interact with software data. These included tasks from data mining task types — clustering, summarising, associating and classifying — and involved activities such as identifying similar data characteristics, distribution and corre-

lations. In §4.4, we showed how a user could interact with a sample software engineering dataset to complete each task using the system. Taggle’s design was intended to cope specifically with the challenges of software engineering datasets, by using appropriate visual mappings available in tag clouds to render the data.

5

Systematic Mapping Study of Tag Cloud Research

Information visualisation techniques can be challenging to evaluate [Plaisant, 2004]. This is because, in addition to general evaluation challenges (such as choosing appropriate questions and tasks, defining methods and then executing the evaluation correctly) the visualisation focus on the data exploration process is difficult to capture and quantify. We embarked on a systematic mapping study of previous research evaluating the tag cloud technique or interactive tools that included tag cloud visualisation. This study identified topics, fields or domains that had not been extensively researched, and approaches and methods which had been used for evaluation. To classify our work, we used a set of information visualisation guiding scenarios which are outlined in §5.1. Research questions and goals for the study are presented in §5.2. The research methodology including data sources, study selection and data extraction are outlined in §5.3. Results for research topic, methods, domains and approaches are discussed in §5.4. Finally,

summary and conclusions are presented in §5.5 and §5.6.

5.1 Strategies for evaluation

In 2012, Lam et al. [2012] identified seven guiding scenarios for information visualisation evaluation. These scenarios were gathered from a systematic review of 803 information visualisation papers (345 of which included evaluations). Of these scenarios, four can be roughly defined as evaluation of the *data analysis process* (EWP, VDAR, CTV, CDA – described in §5.3.3) and the remaining three evaluate the *visualisation use* (UP, UE, VA). These two types of strategies have different goals and use different methodologies.

Evaluation of the *data analysis process* has a goal of understanding the underlying process and roles played by the visualisation itself, and captures a more whole-tool holistic view. The results from this type of analysis may be more meaningful as realistic tasks and scenarios are used. However, results can be more difficult to quantify. Also, the whole tool is evaluated so evaluation may require full featured and mature tool.

The *visualisation use* type strategies do not evaluate the whole tool but a system slice or technique. They are used to evaluate design decisions, explore the design space, benchmark existing systems or test usability. For these strategies, outputs are easier to quantify generated insight. There is a need to break the evaluation into techniques or visual encoding types, so careful prioritisation is needed. Because of this breaking off into sections, more than one experiment may be needed. Tasks may also need to be heavily abstracted which impacts realism.

In the systematic review performed by Lam et al. [2012], only 15 percent of papers used data analysis process type strategies in their evaluation. They concluded that evaluation in the information visualisation sector has been following in the footsteps of evaluations for Human Computer Interaction (HCI) and Computer Graphics (CG), both of which are traditionally focused on controlled experiments and usability evaluations. The data process strategy research questions (such as a tools support for reasoning, knowledge discovery or decision making) are of high relevance and practical value. Lam et al. [2012] highlighted

the need to think critically about the goals of the types of evaluations needed for information visualisation.

We were interested to find out what types of evaluations had been performed for tools utilising tag cloud visualisation techniques, and what research topics and domains the evaluations focused on so we performed a systematic mapping study on 60 selected primary studies from 2007 to 2012.

5.2 Systematic mapping study

We had a number of goals for our systematic mapping study. We wanted to find all papers which have evaluated the effectiveness of the tag cloud visualisation technique in order to identify those areas of tag cloud visualisation which contain either exhaustive research (allowing us to apply and build on), or deserts of information (allowing us to shape future research in this area). Secondly, tag clouds are most commonly associated with the web - we were interested to find out what other domains had proposed and evaluated tag cloud visualisation techniques. Finally, we wanted to discover what evaluation approaches and methods had been used by tag cloud evaluation studies in order to achieve their research goals. Overall, the systematic mapping study should serve to build an overview of what is known about tag clouds as a visualisation technique in general, identifying clusters of evidence and establishing areas of research where knowledge gaps exist.

RQ: *What topics for tag cloud visualisation have been evaluated and to what extent?* We want to establish which topic areas have been focused on in previous research, in order to help shape future research.

RQ: *What evaluation approaches and methods have been used?* We want to discover what types of evaluation have been undertaken for tag cloud visualisation, and what methods were used.

RQ: *For which fields or domains have tag cloud visualisations been evaluated?* The domain which tag clouds are primarily associated with is the web. We

want to know what other fields or domains tag cloud visualisation has been proposed and evaluated for.

5.3 Methods

During the course of the systematic mapping study, the following activities were carried out: define research questions, define data sources and search strategy, perform searches in all designated digital libraries and search engines using the filter, remove duplicate studies (results reported from multiple search engines), review each paper using specified inclusion/exclusion criteria and determine relevance to the topic and research questions, perform data extraction, and perform data synthesis.

5.3.1 Data sources and search strategy

The digital libraries and search engines which were used to extract the articles were ACM digital library¹, IEEE Explore², SpringerLink³, CiteSeer⁴, Scopus⁵, Sage Journals⁶, Scirus⁷, Web of Science⁸, ScienceDirect⁹ and arXiv¹⁰.

The search terms were grouped into three categories 1) pertaining to visualisation technique 2) relating to visualisation type and 3) search terms relating to evaluation.

¹dl.acm.org/

²ieeexplore.ieee.org

³www.springerlink.com

⁴citeseerx.ist.psu.edu/

⁵www.scopus.com

⁶online.sagepub.com/

⁷www.scirus.com

⁸wokinfo.com/

⁹www.sciencedirect.com/

¹⁰arxiv.org/

```
("tag cloud" OR "tag clouds" OR "tagcloud" OR "tagclouds") AND  
("evaluation" OR "qualitative" OR "quantitative"  
OR "experiment" OR "experimentation" OR "experiments"  
OR "study" OR "studies") AND  
("visualisation" OR "visualization" OR  
"user interface" OR "user interfaces")
```

These search groups were joined together with the use of a boolean AND to search document metadata (where available) such as title, abstract, classification and keywords. Additionally, each query had to be adapted according to the interface and query specification of the search engine.

In order to validate the search strategy, a check was performed to ensure a small sample of papers (12) was included in the search results [Bateman et al., 2008; Halvey and Keane, 2007; Hearst and Rosner, 2008; Kaser and Lemire, 2007; Kuo et al., 2007; Lohmann et al., 2009; Oosterman and Cockburn, 2010; Rivadeneira et al., 2007; Schrammel et al., 2009a,b; Seifert et al., 2008; Sinclair and Cardew-Hall, 2008]. These papers had previously been noted as relevant to the research questions during an initial review of the literature.

5.3.2 Primary study selection

Each paper returned from the digital libraries and search engines using the specified query was checked for duplication against other database results. An initial result total of 181 was whittled down to 100 after removal of duplicates (see Table 5.1 on the next page).

Each paper was then checked for relevancy using the title, abstract and keywords. Studies that met one of the following inclusion criteria were included:

- studies describing the evaluation of tag cloud visualisation
- studies describing the evaluation of a visualisation or user interface based on the tag cloud technique
- studies describing the evaluation of a system which utilises tag cloud visualisation

Table 5.1: *Initial search results from digital libraries and corresponding duplicates*

Digital Libraries	Results	Duplicates	Total
ACM digital library	16	1	15
IEEE Explore	38	0	38
SpringerLink	10	0	10
CiteSeer	2	2	0
Scopus	70	55	15
Sage Journals	2	2	0
Scirus	0	0	0
Web of Science	40	19	21
ScienceDirect	2	2	0
arXiv	1	0	1
Totals	181	81	100

- studies describing the evaluation of a system which utilises a visualisation or user interface based on the tag cloud technique

Studies that met one of the following exclusion criteria were excluded:

- studies where only an abstract was available
- studies where the paper was not available in English
- studies describing the evaluation of a system where the evaluation method did not specifically include the tag cloud component
- studies where the described evaluation served as a proof of concept
- duplicate articles of the same study from different sources

In many cases it was not possible to determine relevancy of the study from the abstract alone, particularly in determining whether the evaluation method actually included evaluation of the tag cloud component of a system. For these studies, it was necessary to consider the paper as a whole. Each paper was reviewed twice for inclusion/exclusion criteria, during two passes of the search results as a means of validation. During the second review of the paper, a set of keywords was extracted. These served as the basis for the creation of the classification categories within mapping facets.

Table 5.2: *Search results from digital libraries and corresponding excluded papers*

Digital Libraries	Results	Excluded	Total
ACM digital library	15	1	14
IEEE Explore	38	22	16
SpringerLink	10	4	6
Scopus	15	8	7
Web of Science	21	5	16
arXiv	1	0	1
Totals	100	40	60

During the inclusion/exclusion phase a further 40 documents were excluded from the study, bringing the total number of primary papers included to 60 (see Table 5.2 for the exclusion details, after duplicates had been removed).

5.3.3 Data extraction

For each research question, a set of classification categories was devised within a mapping facet.

For RQ *Research topic*, we determined the categories by extracting keywords from the primary studies:

- evaluating the effectiveness of the tag cloud technique
- determining perceived physical demand or workload
- proposal of evaluation metrics or methodologies
- making design guidelines or recommendations
- determining support for user process (social navigation, incidental learning, reflections of learners, determining credibility of sources, dynamic representation of places/situations)
- discovering limits of visual perception (visual features or properties, layout)
- determining user motivation for use

-
- proposal of a tag cloud enhancement (with respect to relationships, topic clustering, temporal evolution, tag ranking algorithms, tagging interfaces, interfaces for a special dataset or medium, layout optimisation)
 - evaluations of systems/visualisations targeting a special population (Chinese readers, Hebrew readers, tools for the blind)

For RQ *Evaluation approaches and methods*, the Seven Guiding Scenarios for Information Visualisation Evaluation proposed by Lam et al. [2012] were used to classify the evaluation approaches and methods:

EWP: *Understanding Environments and Work Practices.* Studying the design context for visualisation tools including tasks, work environments, and current work practices. Types of research methods include field observation, interviews and laboratory observation.

VDAR: *Evaluating Visual Data Analysis and Reasoning.* Discovering if and how a visualisation tool supports the generation of actionable and relevant knowledge in a given domain. Types of research methods include case studies and controlled experiments.

CTV: *Evaluating Communication Through Visualisation.* Discovering if and how communication can be supported by visualisation (for example through learning, teaching, idea presentation and casual consumption of ambient displays). Types of research methods include controlled experiments and field observation and interviews.

CDA: *Evaluating Collaborative Data Analysis.* Studying whether a tool allows for collaboration, collaborative analysis, and/or collaborative decision-making processes. Types of research methods include heuristic evaluation, log analysis and field or laboratory observation.

UP: *Evaluating User Performance.* Studying if and how specific visualisation features affect objectively measurable user performance. Types of research methods include controlled experiments and field logs.

UE: *Evaluating User Experience.* People’s subjective feedback and opinions. Types of research methods include informal evaluation, usability tests and field observation.

VA: *Evaluating Visualisation Algorithms.* Study the performance and quality of visualisation algorithms by judging the generated output. Types of research methods include algorithmic performance measurement and quality metrics.

By classifying the studies based on these guiding scenarios we get a picture of the underlying evaluation goals, rather than just a description of the type of research methods employed.

For RQ *Visualisation domain*, the categories were again determined by keywords we extracted from the primary studies:

- web (user generated content, database search results, recommendation systems)
- mixed media (image, film, television and audio)
- software engineering
- text corpora
- geographical information
- mobile phone
- digital forensics
- health and medicine (online forums, tool for the blind)
- situated displays
- database search results (OLAP, online database)
- multi-variate data

Papers may cover more than one domain or topic so can be associated with multiple classification types. Where applicable, papers were categorised into sub-topics (as indicated within the brackets).

5.4 Results

The distribution of primary papers over time can be seen in Figure 5.1 — papers span from 2007 to 2012 (the mapping study was conducted 2012). Figure 5.2 shows a bubble chart of the mapping facets research topic and evaluation method. The totals do not match exactly with the total number of papers included in the study as it is possible for papers to cover both multiple topics and use multiple evaluation methods within the study. This is particularly common where studies use the evaluation approach UP (measuring user performance) as this evaluation method tends to use controlled studies and a corresponding UE (user evaluation) component, such as a lab questionnaire requesting subjective user feedback. It is possible to see in this chart a heavy tendency towards UP and UE evaluation approaches within the domains of text corpora and web. The focus on web and text is not surprising as these are the domains where tag clouds are most commonly found.

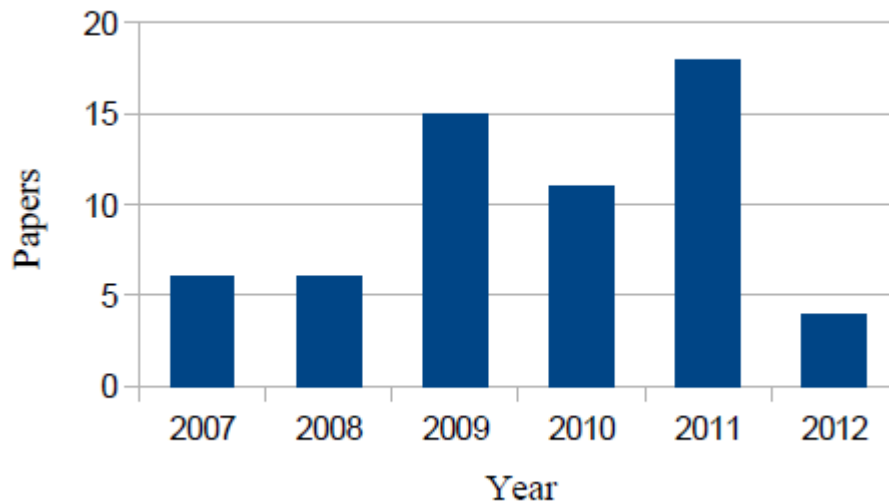


Figure 5.1: *Distribution of papers over time*

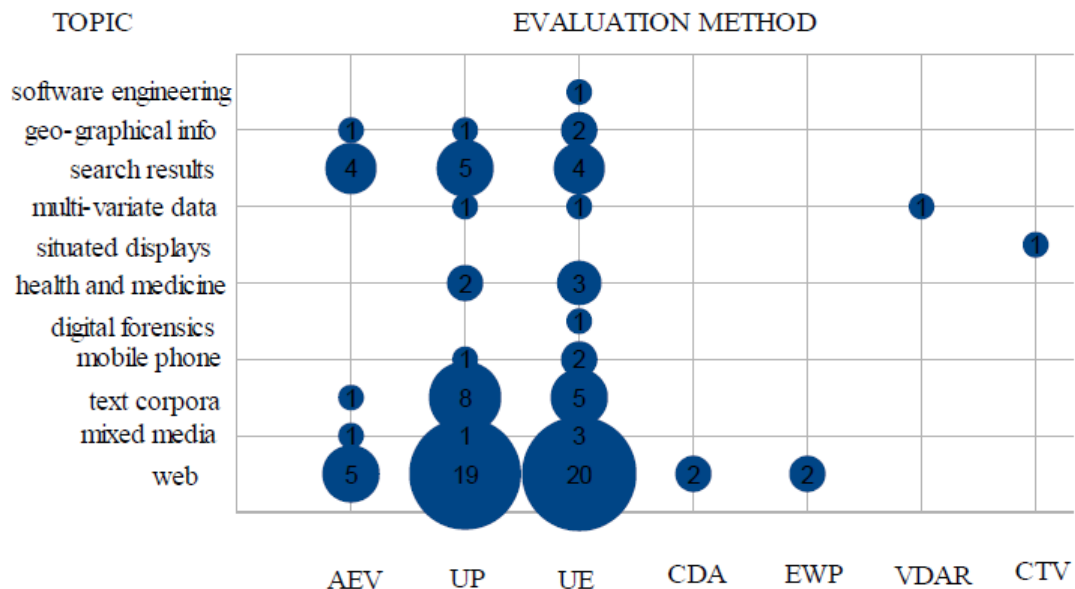


Figure 5.2: *Mapping facets: research topic and evaluation method*

5.4.1 Research topic

A breakdown of the numbers of studies found in each research topic and sub-topic can be found in Table 5.3 on the following page. The research topics were heavily skewed towards evaluations of proposed enhancements to tag clouds (see Figure 5.3 on page 84). The most commonly proposed type of enhancement was to cater for a special dataset or medium [such as [Aras and Huber, 2009](#); [Kim et al., 2009](#); [Kurtz, 2011](#); [Shrinivasan et al., 2009](#)] where interfaces were built for domains such as geo-graphical information, mobile phone, software engineering or multi-variate data. Other popular topics were proposals for improving perceived tag cloud visualisation shortcomings, such as determining relationships and displaying temporal evolution [for example [Caro et al., 2011](#); [Gomez-Aguilar et al., 2011](#)]. Only two papers proposed design guidelines or recommendations [[Bateman et al., 2008](#); [Rivadeneira et al., 2007](#)]. Evaluations of effectiveness and determining the limits of visual perception made up 22 percent of papers. Tag cloud support was researched by 6 percent of papers for a particular user process.

Table 5.3: *Results for the research topic*

Topic	Number	Sub-topic	Number
Proposal of a tag cloud enhancement	41	relationships	7
		topic clustering	6
		temporal evolution	4
		tag ranking algorithms	9
		tagging interfaces	2
		interface for a special dataset	13
		interface for a special medium	2
		layout optimisation	5
Proposal of evaluation metrics or methodologies	2		
Evaluating the effectiveness of the tag cloud technique	9		
Discovering limits of visual perception	8	visual features or properties	5
		layout	3
Making design guidelines or recommendations	2		
Determining user motivation for use	3		
Determining support for user process	6	social navigation	1
		incidental learning	1
		reflections of learners	1
		determine credibility of sources	1
		dynamic representation of places	1
Determining perceived physical demand or workload	2		
Evaluations of systems targeting a special population	4	Chinese readers	2
		Hebrew readers	1
		tools for the blind	1

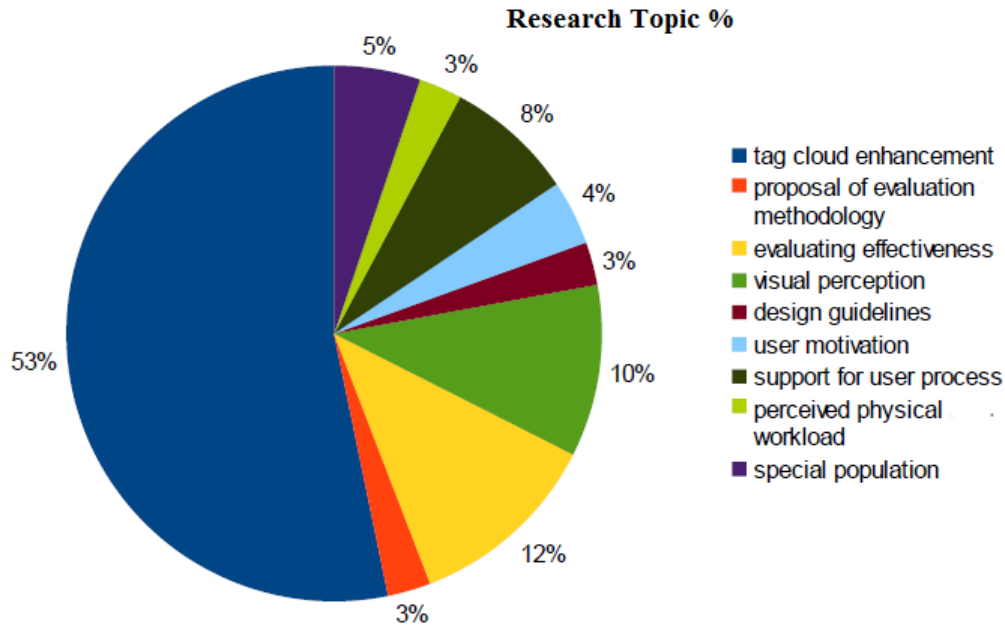


Figure 5.3: *Research topic as a percentage of total papers included in the study*

5.4.2 Research approach and methods

Table 5.4 on the following page shows results for research approach and method. By far the most popular methods of evaluation were the ‘User Performance’ and ‘User Experience’ categories, closely followed by ‘Automated Evaluation of Visualisation’ (for brevity these shall be referred to elsewhere as UP, UE and AEV). These three evaluations strategies make up one of the two main categories of evaluation strategies referred to as ‘visualisation use’. They represent a total of 93 percent of papers surveyed. This is consistent with the findings of Lam et al. [2012], where 85 percent of evaluations in information visualisation papers surveyed were representative of these categories. This was thought to be a possible by-product of the traditions in Human Computer Interaction (HCI) and Computer Graphics (GC) which historically have focused on evaluation by controlled experiment, usability and algorithm evaluation.

All UP category papers reviewed in this study performed controlled experiments, and the vast majority of evaluations within the UE category were carried

Table 5.4: *Results for the research approach and method*

Approach	Number	Method	Number
AEV	14	algorithm performance	13
		quality metrics	5
UP	32	controlled experiments	32
UE	35	informal evaluation	3
		usability test	3
		lab questionnaire	30
CDA	2	log analysis	2
EWP	2	interviews	2
VDAR	1	case study	1
CTV	1	field observation	1
		interviews	1

out via lab questionnaires.

5.4.3 Visualisation domain

Visualisation domain category results are found in Table 5.5 on the next page. Nearly half of all visualisation domains pertained to the web, with a significant portion of those relating to user generated content [for example Bateman et al., 2008; Halvey and Keane, 2007; Kaser and Lemire, 2007; Skoutas and Alrifai, 2011]. Furthermore, another 11 percent of all domains evaluated visualisations of text corpora. This is understandable given tag cloud visualisation web-based beginnings, and visualisation of label identifiers and textual data is a key advantage of tag clouds. However, it should be possible to apply an information visualisation technique such as this to any domain where textual data exists. Database search results, particularly for online databases, were another popular domain of research [Wilson and Wilson, 2011; Yamamoto et al., 2009].

5.5 Summary and discussion

We wanted to build an overview of what was known about tag clouds as a visualisation technique in general, identifying clusters of evidence and establishing areas of research where knowledge gaps existed. We identified 60 papers spanning from

Table 5.5: *Results for the visualisation domain*

Domain	Number	Sub-domain	Number
Web	33	user generated content	25
		database search results	7
		recommendation systems	1
Mixed media	4	image	1
		film	1
		television	1
		audio	1
Text corpora	9		
Mobile phone	2		
Digital forensics	1		
Health and medicine	3	online forums	1
		tools for the blind	1
Situated displays	1		
Multi-variate data	2		
Database search results	9	online database	7
		OLAP	2
Geographical information	3		
Situated displays	1		
Software engineering	1		

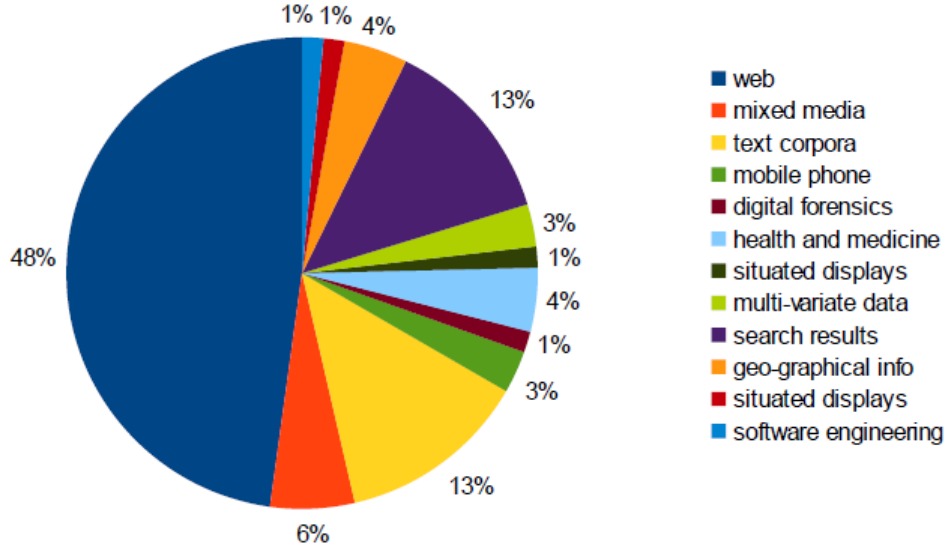


Figure 5.4: *Visualisation domain as a percentage of total papers included in the study*

2007 to 2012 which were relevant to this evaluation of tag cloud visualisation evaluation.

RQ: *What topics for tag cloud visualisation have been evaluated and to what extent?* Topics and total number of papers are found in Table 5.3 on page 83. We wanted to find all papers which have evaluated the effectiveness of the tag cloud visualisation technique so we could identify relevant research that might be applied and built upon, or areas where information was sparse. Only nine papers evaluated the effectiveness of tag cloud visualisation. While this can be widened to 16 to include papers which discuss issues surrounding the visual perception of tag clouds, this indicates there is still room to define the overall effectiveness of tag clouds as a technique. There was a large proportion of papers (43 percent) evaluating interactive interfaces for special datasets, mediums or populations.

RQ: *What evaluation approaches and methods have been used?* Evaluation approaches and total number of papers are found in Table 5.4 on page 85. We wanted to discover what types of evaluation have been undertaken for tag

cloud visualisation, and what methods were used. The vast majority (93 percent) of research performed evaluations relating to ‘User Performance’, ‘User Experience’ and ‘Automated Evaluation of Visualisation’ – *visualisation use* type categories. Within these categories, the methods of evaluation included controlled experiments, lab questionnaires and automated algorithm performance measurements.

RQ: *For which fields or domains have tag cloud visualisations been evaluated?*

Domains and total number of papers are found in Table 5.5 on page 86. The domain which tag clouds are primarily associated with is the web. We wanted to know what other fields or domains tag cloud visualisation had been proposed and evaluated for. The surveyed research indicated a majority of papers (48 percent) were researching tag cloud visualisation for the web and user generated data domain. This is understandable and stems from the initial beginnings of tag cloud visualisation on the web. However, recent research in domains such as mobile phones, digital forensics, and health and medicine indicate researchers are beginning to consider the viability of tag cloud visualisation in other areas [such as [Aras and Huber, 2009](#); [Jankun-Kelly et al., 2011](#); [O’Grady et al., 2012](#)]. In the software engineering domain, there has been one evaluative study utilising tag clouds [[Kurtz, 2011](#)].

The results in this systematic mapping study match those discovered by [Lam et al. \[2012\]](#) where information visualisation evaluations methods focus primarily on controlled experiments and lab questionnaires within approaches UP, UE and AEV. This is despite 43 percent of papers evaluating interactive interfaces to explore data or discover information for special datasets, mediums or populations. Other approaches to evaluation may need to be considered to cover a wider variety of research goals.

5.6 Conclusion

The surveyed research indicates a strong prevalence in the research for the web and user generated data domain with software engineering focused on in only

one paper. Tag cloud visualisation itself has not been as extensively evaluated as other areas, indicating there is still room to define their overall effectiveness and develop ways to improve the tag cloud as a technique. A large proportion of papers evaluated interactive interfaces tailored to particular datasets, populations or mediums. We should note that no interface identified in the mapping study proposed a system such as Taggle, where data fields from a multi-variate dataset are mapped to tag cloud visual properties and manipulated interactively. Moreover, despite the prevalence of interactive interfaces, evaluation approaches were of a limited range — predominantly *visualisation use* techniques measuring user responses times, as opposed to strategies that consider the *data analysis process*, which are of high relevance and value when evaluating tools with data exploration and knowledge discovery goals.

In recent years there has been a spate of research surrounding tag cloud visualisation. This systematic study of 60 papers (from 2007 to 2012) was undertaken in order to discover what sorts of topics relating to tag cloud visualisation have been evaluated and to what extent. This work provided an overview of what is known about tag clouds, and helped us plan and focus our overall evaluation strategy which is presented in Chapter 6.

6

Evaluation Strategy for Taggle

Our systematic mapping study of tag cloud research (presented in Chapter 5) identified tag cloud visualisation as a technique had not been as extensively evaluated as other topics (such as interactive interfaces incorporating tag clouds), indicating opportunities for measuring effectiveness and developing ways to improve the tag cloud approach. No research identified in the mapping study described a system with interactive features like Taggle, where data fields from a multi-variate dataset are mapped to tag cloud visual properties. There were a limited range of evaluation approaches despite the widespread presence of interactive interfaces, typically utilising user performance measurements (*visualisation use* type techniques), rather than strategies considering the *data analysis process*, which are of high practical value and relevance when evaluating systems to explore data and discover information. We were therefore encouraged in several aspects: by focusing on software engineering we were promoting use of tag cloud visualisation outside the typical web and user generated data domains; that our tool was unique in its approach, and that to conduct an evaluation with broad-ranging

goals we should consider multiple targeted evaluation strategies.

Based on the results of the systematic study, this chapter outlines the process and outcomes of designing an evaluation plan for Taggle. We describe mapping out our overall evaluation strategy and associated methodologies in §6.1, and evaluations that were selected to be conducted in §6.2. Generation of suitable datasets for experimentation is discussed in §6.3. In §6.4 we describe the process we took to conduct three of our experiments on an eye-tracking machine.

6.1 Overall evaluation strategy

Our overall goal for evaluation was to explore the potentials and limitations of our tag cloud visualisation tool Taggle. We decided to map out sample areas of the tool which were of interest, in order to strategically plan a broad-ranging series of evaluations. This map is represented in Figure 6.1 on the next page. Potential areas of evaluation were divided into the part of the system being looked at — *visual encoding* or *interactive tool*. Each potential research question was associated with an appropriate evaluation strategy (highlighted in blue): *User Performance*, *User Experience*, *Visual Data Analysis and Reasoning* (for more details of strategies identified by Lam et al. [2012] see Chapter 5). For each design choice or evaluation strategy, sample methodologies are outlined (presented in a white cloud).



Figure 6.1: *Evaluation strategy map for an interactive tag cloud visualisation tool*

In the green box on the map under the *visual encoding* section, there are a list of *design choices* that needed to be made. Selection of approaches could be made either through controlled experiments comparing choices, or relevant information found in previously defined guidelines, research or design principles. Questions regarding the visual encoding of the tool that are suited to evaluation by *user performance* are divided into understanding the limits of human visual perception and cognition, or involving a comparison of interaction or visual encoding techniques. These questions can be answered by experimentation if no previous research exists. Additionally, visual encoding of the tool may benefit from evaluation of *user experience*, where questions are answered through subjective user questionnaires or the process of heuristic evaluation.

In the red box section describing research questions for the *interactive tool*, there is a second set of *user experience* questions which may be answered by heuristic evaluation or laboratory questionnaire. An *evaluate visual data analysis and reasoning* section defines research questions related to decision making and knowledge discovery, where responses can be gathered from case studies or user experiments with the interactive tool.

Ultimately, the creation of this map was a highly worthwhile experience as it allowed us to select the most appropriate evaluation methodologies for the research questions in which we were interested.

6.2 Selected evaluations

Many of the design choices that were possible for the visual encoding of the tool had been made according to the design considerations outlined in Chapter 3 which were informed by guidelines or other research. A decision was made to conduct user experiments comparing the enhanced tag cloud features which were included in Taggle with tag clouds generated in a more conventional fashion. Two experiments were designed for the following special features:

- the use of background colour to increase the colour space of the tag text
- the use of multiple visual features to highlight and reinforce data mappings

Use of background colour Small tags in Taggle (or any tag cloud visualisation) are inevitable if we are using a full range of tag sizes. In software engineering we are dealing with lots of tags due to the large dataset sizes, and we need to know if users can identify the variables mapped to colour, should they be displayed with smaller tags (or Taggle’s small iconified tags). Use of background colour — which increases the amount of colour space of the tag text — may help here.

Use of multiple visual mappings We think mapping multiple visual features may be helpful for users to identify data correlations, through highlighting and reinforcing selected data mappings. Reinforcing these all important mappings may also assist users to explore the data more effectively.

These experiments are a good starting point in our evaluation plan, laying the foundations for more specific targeting of other areas of Taggle’s visual encoding in future. Two different layouts were also tested in each experiment. The experiments compared user visual search response times (a *user performance* evaluation strategy) and were conducted on an eye-tracking machine. Details can be found in Chapters 8 and 9.

An evaluation with domain experts using a special heuristic set for information visualisation was planned to elicit user experience feedback on both the visual encoding and interactive tool. Subjective user feedback was sought to clarify research questions regarding the comprehensibility of the visualisation technique, as well as assessing the system usability. Outcomes of this evaluation are discussed in Chapter 7.

A *visual data analysis and reasoning* experiment was conducted to try to discover if (and how) Taggle supports data exploration for a software engineering dataset. This was a whole-tool approach focused on the knowledge discovery process. Chapter 10 details the executed experiment and results.

6.3 Dataset generation

Previous tag cloud research has used a variety of sources for experiments requiring tag corpora. In general, for quantitative experiments the tag labels themselves were taken from real life data such as psycholinguistic databases [Bateman et al., 2008; Rivadeneira et al., 2007], online sources such as Flickr, Zoomclouds [Kaser and Lemire, 2007; Schrammel et al., 2009b] or contained encyclopaedic-type categorical data (unspecified sources) [Halvey and Keane, 2007; Oosterman and Cockburn, 2010] while the tag weightings were generated artificially to match experiment goals. These experiments focused on datasets with a more conventional tag cloud application in mind (web site content visualisation for example), and were using only one weighting. Creation of tag corpora for a multi-variate dataset is a more complex affair as we are concerned with multiple variables. For realism in tasks both general and in the software engineering domain, we need the data to make sense, with variables containing correlations and realistic distributions. In addition, we need to avoid user bias which might occur in datasets where users could be expected to have some personal knowledge.

We required two types of datasets in our experiments. For the *visualisation use* experiments (Chapters 8 and 9), we wanted to use datasets from a general domain (to widen both the pool of participants and potential datasets) containing categorical data with a nominal data field which could be used as a text label for tags. Like previous research, we chose to source the textual data fields from real life data and selected lesser-known knowledge areas to minimise bias. Data for numeric variables in the same dataset were generated artificially. For the second type of dataset for use in the *data analysis process* evaluation, we required a software engineering dataset. This was generated in the same way as the general domain datasets, with class names taken from an open source project and software metrics generated artificially. See Figure 6.2 on the following page for example datasets compiled from both real life data and generated data.

It also would have been possible in many cases to have used a subset of the original data rather than artificial generation, but this would have required careful analysis first to discover outliers/distributions and other special features. Taking a subset may have affected the dataset features in unintended ways whereas

artificial data generation gave absolute control over the correlations and features of the dataset.



Figure 6.2: Examples of datasets compiled from both real life and artificially generated data

6.4 Conducting an eye-tracking experiment

Experiments in Chapters 8, 9 and 10 were performed using an eye-tracking machine. Eye-tracking is the technique used to capture and measure eye movements, allowing analysis to be performed on the patterns of visual attention of a user performing a series of specific tasks. From these analyses, inferences can be made about the user’s cognitive processes [Olsen et al., 2010]. Eye-movement is typi-

cally divided into two types 1) saccades (quick movements of the eye from one location to another) and 2) fixations (cessation of the eye movement, pausing on an object of interest for a period of time). Eye-tracking data is collected and then interpreted through visual representations such as gaze plots (a presentation of saccades between fixations, showing the eye scan path), or heat maps showing time spent at each fixation (example Figure 8.2 on page 129 for an example of a gaze plot).

Experiments were performed using a Tobii T60 eye-tracker. This has an optimal operating distance of 65 cm, so the participant must be placed approximately this far away from the eye-tracker. The T60 can capture stimuli at a maximum radius of 35 degrees and is integrated with a high resolution 17" monitor.

Figure 6.3 on the following page shows the pipeline process which was followed using the Tobii studio software to complete experiment design and recording through to statistical data exportation. Tobii studio was used to input the experimental design with selected test media and manage the participants and experiment counter-balancing. During the experiment process, participant eye-movement was recorded. Areas of interest (target tags) were defined within the tag cloud images and the software collected metrics regarding those areas of interest (such as the time to first fixation). The software was used to play back video recordings with eye-movements, to manually analyse eye movement search patterns within the media.

6.5 Summary and discussion

Our systematic mapping study of tag cloud research showed us that to broadly evaluate Taggle we should consider multiple targeted evaluation strategies. To that end, we followed a procedure of mapping out our overall evaluation strategy based on evaluation approaches presented by Lam et al. [2012]. From this map we selected evaluations and appropriate methodologies. Experimentation conducted from this map required generation of realistic datasets sourced from both general and software engineering domains. Three of our experiments were run on eye-tracking machines, allowing us to perform analysis on patterns of user visual attention. The process of conducting an eye-tracking experiment also required

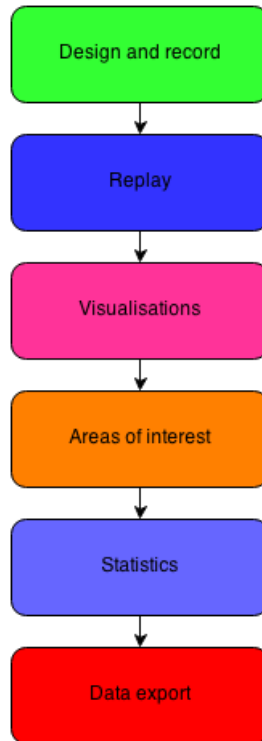


Figure 6.3: *Pipeline of an eye-tracking experiment*

carefully management and adherence to a pipeline process from experiment design to statistical data exportation.

7

Heuristic Evaluation of Taggle

An inexpensive and popular method for evaluating the usability of an interactive tool is to employ an evaluation using a set of heuristics such as Nielsen’s *Ten Usability Heuristics* for user interface design [Nielsen, 1992]. In this type of evaluation, a number of experts review a tool and determine how well it follows some predefined guidelines. As part of our overall evaluation strategy, we conducted a heuristic evaluation of interactive tag cloud visualisation tool Taggle using a heuristic set specially designed for information visualisation tools. Subjective user feedback was sought to clarify research questions regarding the comprehensibility of the visualisation technique, as well as assessing the system usability. In §7.1, we outline the set of heuristics used to evaluate Taggle. The methodology of the evaluation is detailed from §7.2 through §7.5. Results from the heuristics themselves, the guideline checklist and questionnaire are presented in §7.6. Finally, subsequent adaptations resulting from the evaluation are presented and summarised in §7.7 and §7.8.

7.1 Heuristics

Heuristic evaluation is intended as a “discount usability engineering” method as opposed to an expensive user trial, in particular because so few participants are needed. Nielsen’s recommendation was to use three to five evaluators since that number is sufficient to identify most of the issues.

Information visualisation tools require a set of heuristics specifically tailored to finding usability issues that focus on the process of data exploration and visualisation techniques. Existing research has identified a number of heuristic sets and guidelines. The well-known ‘information seeking mantra’ by [Shneiderman \[1996\]](#) guides successful data exploration. [Luzzardi et al. \[2004\]](#) proposed an extended set of ergonomic criteria for information visualisation techniques which were designed to assess both visualisation and interactivity techniques for hierarchical data representations. [Zuk and Carpendale \[2006\]](#) outlined a set of heuristics based on previous theories and principles in perception and cognition by [Bertin \[1983\]](#), [Tufte \[1990\]](#) and [Ware \[2004\]](#). Finally, [Forsell and Johansson \[2010\]](#) took these and other developed heuristic sets such as [\[Nielsen, 1992\]](#) and used empirical methods to synthesize them into a new set of ten heuristics which could provide the widest possible explanatory cover proportionally for all 63 heuristics presented in their study. It is this set of heuristics which was used to evaluate Taggle. Artefacts used in the heuristic evaluation sessions (including a complete description of the heuristics with additional guidelines on how to apply them) are available in [Appendix B](#).

The purpose of the heuristic evaluation was to elicit user feedback which resulted in design refinements, and to check the tool was sufficiently detailed to be used in further trials. Subjective feedback was gained on certain points of interest to help shape tasks for future experiments. In particular, we wished to clarify the following research questions:

RQ: Is the visualisation technique itself instinctively comprehensible? Can the visualisation supply the user with a good overall picture of the data?

RQ: Can users infer general information about the data from interacting with the visualisation? What kind of information?

RQ: Is the data mapping process understandable?

RQ: Is the supplied system interactivity sufficient for accumulating knowledge about the data?

By evaluating our software through the use of a heuristic set, we wished to elicit user opinion on the system usability with regards to visual representation, interactivity, flexibility and consistency in design choices.

7.2 Participants

Three evaluators were recruited from Lincoln University and Canterbury University (two female and one male, two academic staff from the Department of Applied Computing and one graduate student from the Department of Computer Science). All evaluators had a solid background in computer science or human computer interaction, and had gained practical and theoretical experience in the field of usability engineering.

7.3 Apparatus

The evaluations were conducted with the Taggle prototype on a Windows XP PC with 2GB RAM. The 20-inch LCD screen had a resolution of 1680×1050 pixels.

7.4 Tag corpora

Table [7.1 on the next page](#) describes the datasets used in the heuristic evaluation. Four datasets were developed, two generic domain datasets and two software engineering. Text fields were sourced from real datasets found online, and in some cases numeric data was artificially generated.

7.5 Procedure

Each evaluator completed the heuristic evaluation in a separate session which followed the process outlined below:

Table 7.1: *Datasets used in heuristic evaluation*

Data	Columns	Description
Baby name popularity	Name (text) US popularity (number) NZ popularity (number)	The popularity rankings are a number between 1–75 representing the popularity ranking of that particular baby name in a given country. For the 75 names represented, the name ranked number 75 is the most popular, and the name ranked number 1 is the least popular.
Software metrics	Java class name (text) 12 software engineering metrics (numeric)	The software being measured is the open source project jUnit. Twelve metrics have calculated per class, generated by the Sonar software quality platform.
Agile development stories	Title (text) Story (text) Actor (text) Estimate in hours (number)	Describing 35 stories for use in an agile development project for creation of an book store management system. A time figure given in hours has been estimated for the completion of each story.
Cross country challenge	Runner ID (number) Surname (text) Club name (text) Score (number)	The surname and club for 100 runners in a cross country challenge. Each ID pertains to a runner. There are five possible clubs and runners are associated with one club only. The lower score represents a higher placing in the race.

-
1. Training — tag clouds and data exploration (5 minutes)
 2. Taggle guided navigation (10 minutes)
 3. Heuristic overview (5 minutes)
 4. Free exploration and issue sheet fill-out (30 minutes)
 5. Heuristic checklist and questionnaire (10 minutes)

Pre-evaluation, five minutes of training describing tag cloud visualisation using both generic domain and software engineering datasets was given (see Figure 7.1 for an example training image used). Evaluators then completed a short tutorial on the Taggle software — using the same generic dataset, the user was guided through a tour of various interactive features of the software. Participants were given a full and complete description of each heuristic including guidelines on how to apply them (§B.1) and these were then outlined in brief with the opportunity for the evaluator to ask questions.



Figure 7.1: *Training image: tag cloud visualising the New Zealand national anthem. Size and transparency visual properties are used to display word frequency of anthem verses. Stop words such as ‘is’ and ‘the’ have been ignored.*

Following the training sessions, the evaluators were then asked to freely explore Taggle to try to discover information about one or more of the provided datasets. As they explored Taggle, they completed an issue sheet detailing each

usability problem found and the severity level. Each evaluator was observed throughout the session and notes taken regarding their comments and suggestions. Evaluators were told to look at any number of the datasets, whichever contained data that was more interesting to them. To take pressure off the analysis, they were also informed that some datasets contained a correlation between data variables, showed an obvious skewed distribution in the data, or contained obvious outliers and that some datasets did not — there was no ‘right answer’ or any specific data pattern to look for.

Evaluators were also given a sheet of Taggle typical usage patterns (can be found in §B.3) that provided a set of sample tasks for exploration of a dataset. These matched the exact steps taken in the guided navigation tutorial and included steps to set up data mappings, interact with and filter the data, and explore the data to generate hypotheses.

When the evaluator felt that they had learned all they could regarding the chosen dataset or datasets, they then completed a checklist where the evaluator could mark off whether they felt the tool had sufficiently or partially met the recommended guideline for each heuristic. Following this, a questionnaire was completed where evaluators were asked to rate the tool with regards to comprehensibility and knowledge discovery support on a 5-point Likert scale (from 1=“Strongly disagree” to 5=“Strongly agree”).

7.6 Results

7.6.1 Heuristics — issues, comments and observations

A complete and full list of usability issues found mapped to the appropriate heuristic with evaluator assigned severity level can be found in §B.4. Figure 7.2 on the next page shows the number of issues found per heuristic. This doesn’t reflect the total number of issues found, as some issues are related to multiple heuristics.

Initial presentation of dataset Evaluators thought that the initial starting point when exploring unknown data was confusing. Some users felt that it would

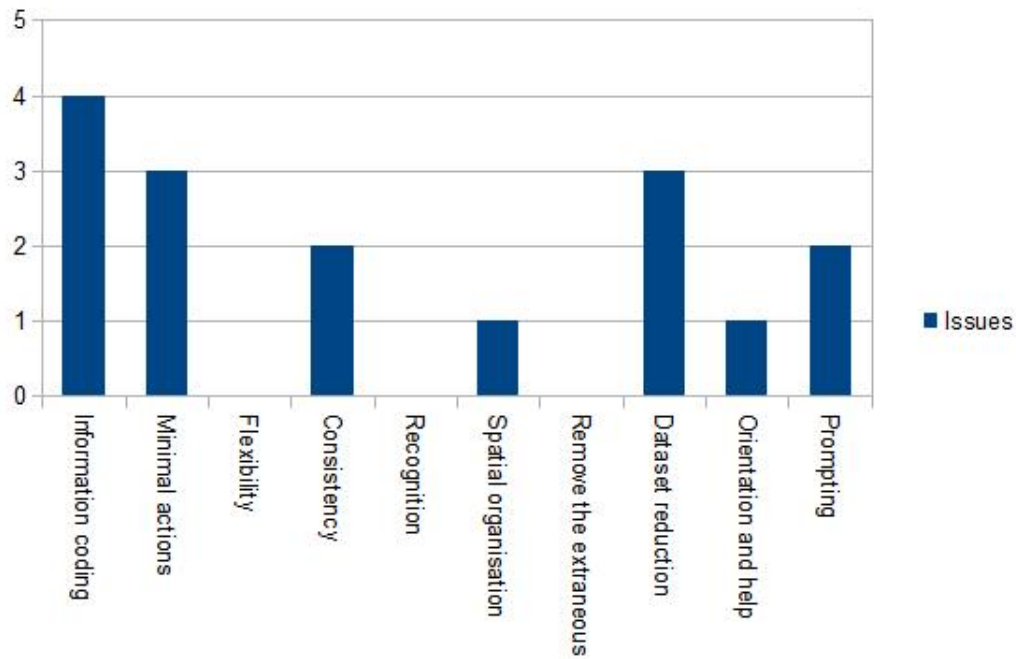


Figure 7.2: *Number of issues per heuristic (issues may be related to more than one heuristic)*

be helpful to have the cloud pre-mapped to data fields and a starting cloud displayed on initiation of the software. The default mappings select all options except for transparency. Users felt that starting with just one mapping and adding others as needed was more useful to build knowledge about the dataset.

Selection of data mappings The mapping selections were perceived to be the major area which needed improvement. Although not given specific tasks, users quickly ascertained what useful questions might be asked of a dataset (e.g. what club has the best cross country runners?), but had some difficulty selecting mappings that might help them answer these questions. What was useful for decision making in mapping selection was an overview of each variable as a starting point for further investigation (e.g. what is the data range for this variable? Is it text or numerical? Is it categorical?). This information had been provided for them on

paper (Table 7.1 on page 102), but users commented it would be useful information to integrate into the tool. As a result of initial misunderstandings over data field type, users were sometimes confused by mapping variables to textual data instead of numerical data (this is not useful for mapping to some visual properties such as font size). Conversely, evaluators didn't always map the text label to the tag text but used an numeric ID column. While not as immediately/obviously useful as mapping text, this did have the added advantages of maximising space efficiency in the visualisation, minimising the overemphasising effect of long text and allowing the user to gain an overview of the data in the ID column (ranges for example).

Users didn't use the transparency mapping much at all — it wasn't perceived to be as useful as font size or colour, and some users were confused by the transparency mapping interface. The drop-down box for layout was confusing as it contained layout options that were for layout research purposes. Evaluators didn't change the default colour set much, although one user commented that black and white were the most helpful colour combinations. A suggestion was made for data to be more complemented by the colour legend. It would be useful for categorical data to be displayed as a colour for every category in a data field.

Another feature requested for mapping selection was the reversal of the mappings properties. This was available for the ordering property but not for size or colour. An evaluator noted that data wasn't always best displayed from smallest to largest numeric value.

Static and dynamic filtering Most time was spent by users changing mappings rather than filtering data. One user remarked that this was because of the free exploration nature of the evaluation — if we had asked more specific questions about the dataset, then more time would have been spent filtering the data to a more appropriate subset according to the question.

The mouse wheel filter was a dynamic filter (appearing on static tab). It was suggested that it would be more useful as a static filter — useful to discover what settings are most appropriate to display — and should stay observed after the reset/update button click. This option is not known to the user until they select the filter tab. The dynamic filtering interface was problematic for one user as

they didn't realise they could select the knobs on the slider.

Visual encoding The visualisation itself didn't pose too many problems for the users. One user wasn't aware of the overview right click pop-up because of an oversight in the training. A complaint about typewriter layout was that it created a disjoint in the flow of the data field mapped to order at the end of each row.

7.6.2 Heuristics — guideline checklist

The guideline checklist can be found in §B.2. Evaluators were asked to mark each guideline specifying whether they considered each criteria fulfilled, not fulfilled, or partially fulfilled. Each figure visualising results of a heuristic presents marks averaged from evaluators on a scale between zero and one (where zero means the criteria is not fulfilled, and one means the criteria is fulfilled). This section summarises and discusses guideline checklist results for each heuristic.

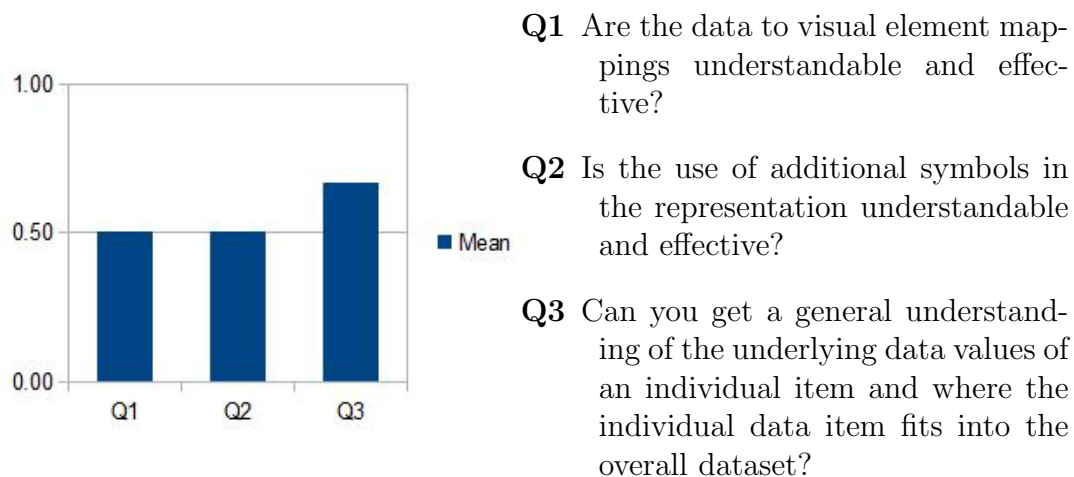


Figure 7.3: *Information coding — visual representation*

Information coding — visual representation (Figure 7.3) Evaluators felt this criterion was partially fulfilled. One evaluator felt that an overall understanding of the underlying dataset was only possible after a user had some experience

or practice with the system. We felt that the responses to this guideline reflected the problems evaluators found with the mapping selection interface.

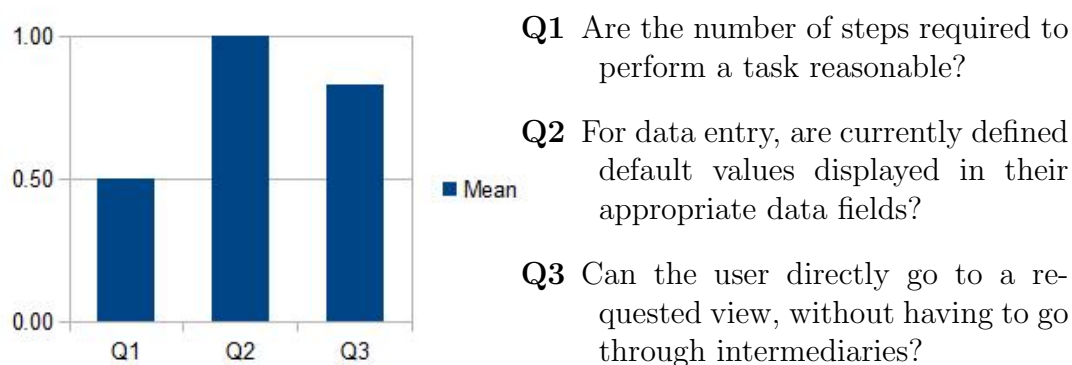


Figure 7.4: *Minimal actions — interactivity*

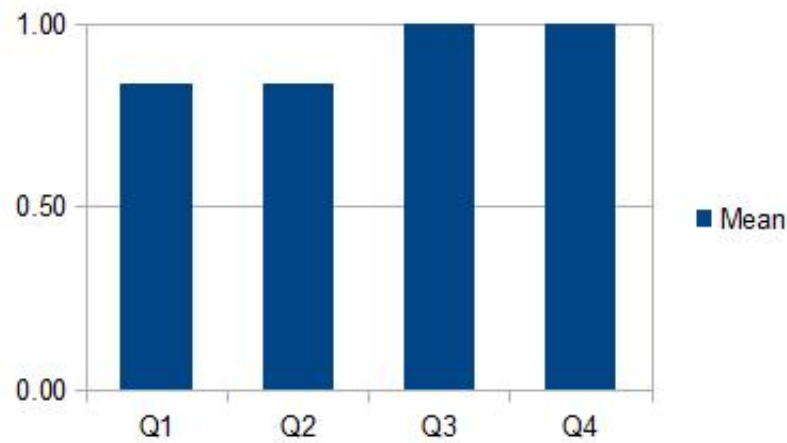
Minimal actions — interactivity (Figure 7.4) This criterion was generally satisfied. Issues regarding the number of steps required to perform a task were interpreted as relating to default mapping values and the initial display of the dataset.

Flexibility — interactivity (Figure 7.5 on the next page) Evaluators were satisfied that the system provided sufficient flexibility in controlling tag cloud configuration.

Consistency — interactivity (Figure 7.6 on page 110) Criterion was satisfied for consistency in user interface — configuration controls, window titles, labels and general task procedures.

Recognition rather than recall — interactivity (Figure 7.7 on page 110) This criterion was generally satisfied although evaluators wanted greater access to summary information relating to the dataset.

Spatial organisation — visual representation (Figure 7.8 on page 111) Evaluators rated Taggle fairly well for this criteria. One comment regarding the



Q1 Are users provided with sufficient means to control visualisation configuration?

Q2 Are users permitted to define, change or remove default values for settings?

Q3 When some displays are unnecessary, can users remove or hide them temporarily?

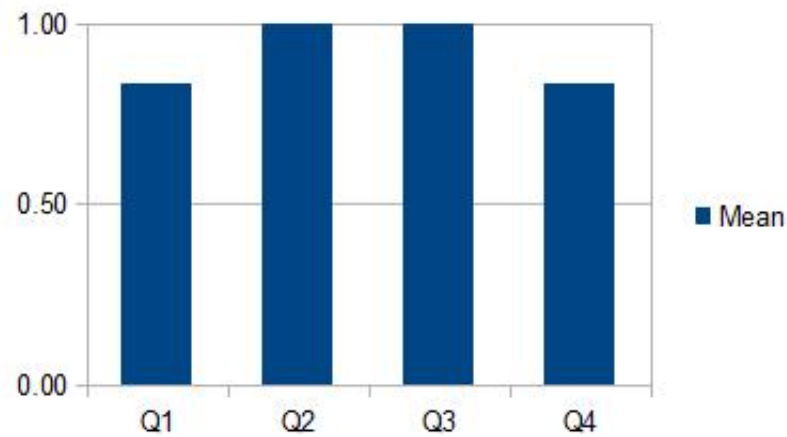
Q4 Can users change settings in any order?

Figure 7.5: *Flexibility — interactivity*

spatial aspect of the visualisation was that the tag cloud could sometimes be very small in the middle of the canvas which wasn't an efficient use of space. This depended on the data mapped though. Another evaluator commented that layout may or may not follow a logical organisation because it depended on their own appropriate mapping selections.

Remove the extraneous — interactivity (Figure 7.9 on page 112) This criteria was marked as partially met. Evaluators felt that it was important to get a simplified overview of the dataset initially which hadn't been provided for in the interface.

Dataset reduction — interactivity (Figure 7.10 on page 112) Evaluators were satisfied that the system provided sufficient means for filtering and reducing



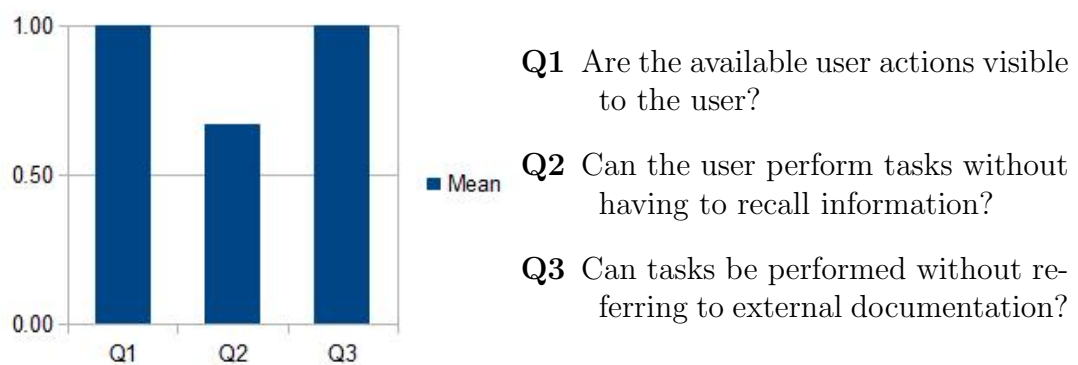
Q1 Are window titles always located in the same place?

Q2 Are the configuration controls consistent?

Q3 Are similar procedures used to perform tasks?

Q4 Are labels (phrasing, punctuation, placement) consistent?

Figure 7.6: *Consistency — interactivity*



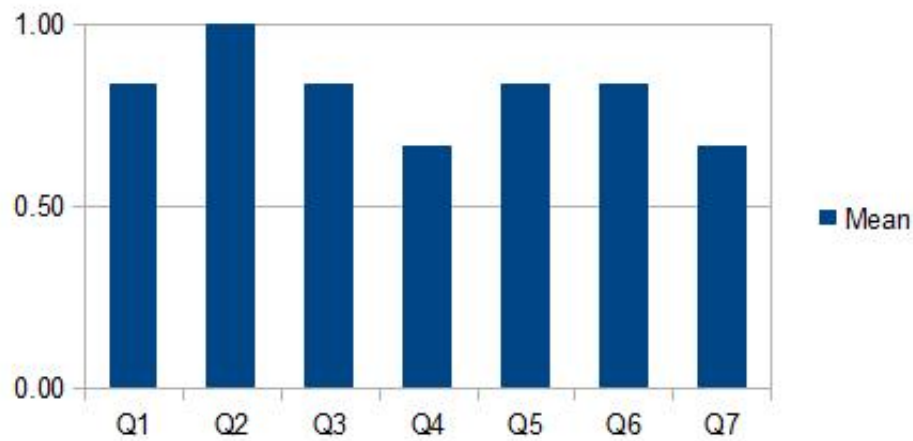
Q1 Are the available user actions visible to the user?

Q2 Can the user perform tasks without having to recall information?

Q3 Can tasks be performed without referring to external documentation?

Figure 7.7: *Recognition rather than recall — interactivity*

the dataset.



Q1 Can I easily locate an information element in the display?

Q2 Are the individual elements legible?

Q3 Is space used efficiently in the layout?

Q4 Am I aware of the overall distribution of information elements in the representation?

Q5 Are some objects occluded by others?

Q6 Does the layout follow a logical organisation?

Q7 Do I understand how tag placement is related to the selected layout and ordering?

Figure 7.8: *Spatial organisation — visual representation*

Orientation and help — interactivity (Figure 7.11 on page 113) Evaluators felt help interactivity was only partially satisfied. More assistance could be provided to orientate the user to the system through means such as redo and undo.

Prompting — interactivity (Figure 7.12 on page 114) As with orientation and help, evaluators felt the prompting status of Taggle could be improved with the addition of things such as online help and more cues on acceptable data entries. We feel manuals and online help guides are more useful and expected for

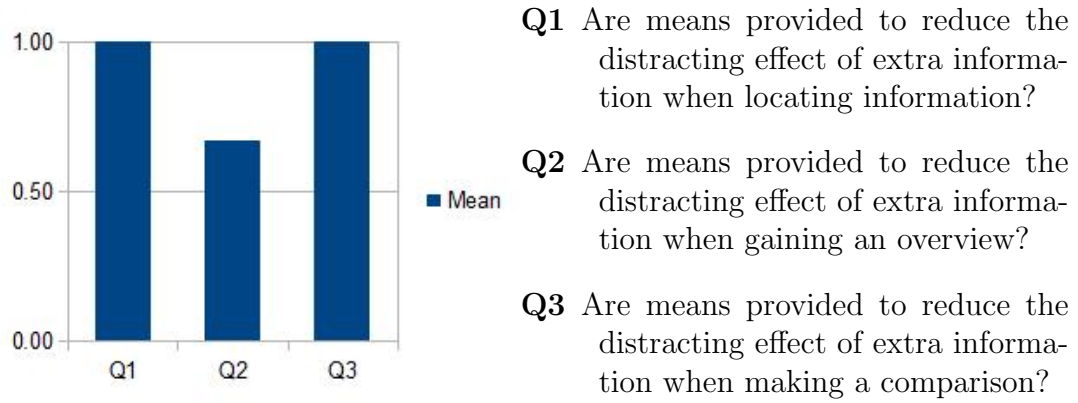


Figure 7.9: *Remove the extraneous — interactivity*

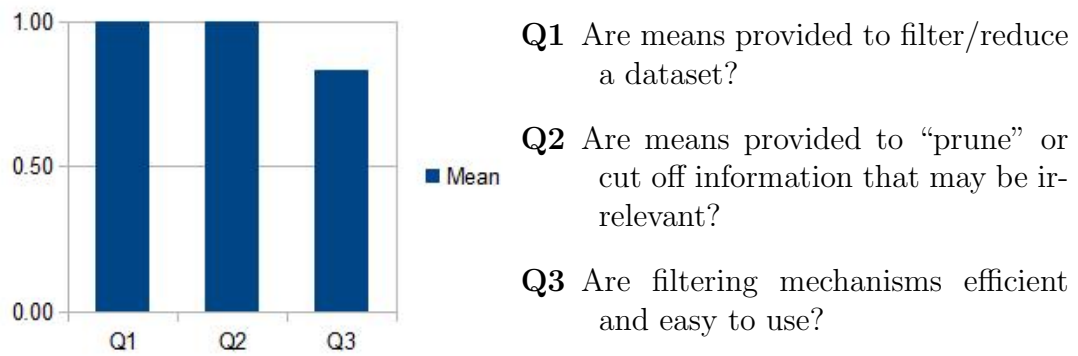


Figure 7.10: *Dataset reduction — interactivity*

a mature tool rather than a prototype.

7.6.3 Questionnaire

This section presents results for a 5-point Likert scale questionnaire where evaluators were asked to rate the tool with regards to comprehensibility, knowledge discovery support, and other research questions from 1 (“Strongly disagree”) to 5 (“Strongly agree”). Results for each questionnaire section are summarised and discussed.

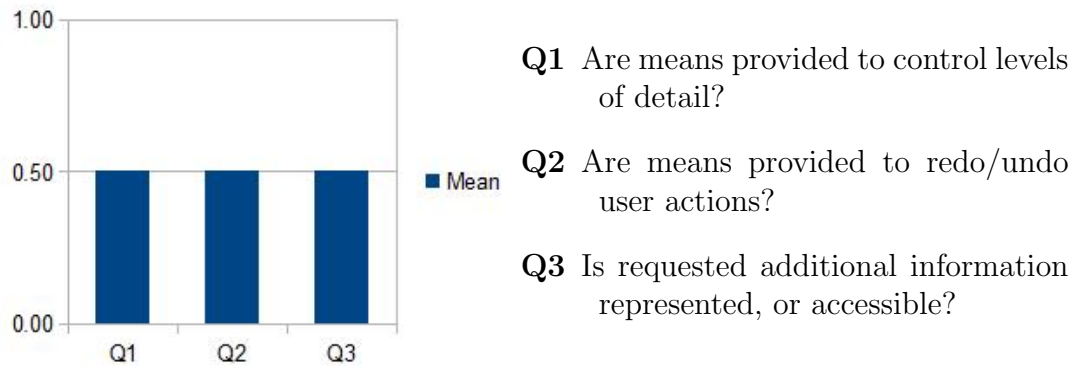
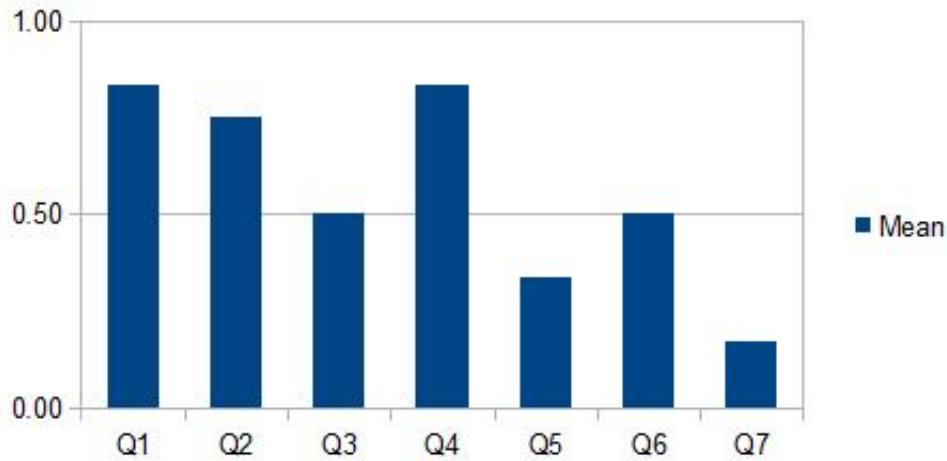


Figure 7.11: *Orientation and help — interactivity*

Is the visualisation technique instinctively comprehensible? (Figure 7.13 on page 115) Users agreed or strongly agreed that they could immediately comprehend that tags with contrasting font sizes, colours or transparencies had differing underlying data values. It was observed that evaluators frequently mapped the tag text to numeric fields. This may have contributed to the feeling that the tag label didn't always give an immediate visual clue to the represented data, as this greatly depends on the users choice of mappings.

Are users able to infer general information about the data from interacting with the visualisation? What kind of information? (Figure 7.14 on page 116) All users were able to infer the underlying data distribution (ranging from one dataset to all datasets), although with various levels of difficulty. All users were able to identifier outliers in a dataset and distinguish tags which had similar data characteristics. One evaluator commented they found it difficult to choose the right mappings when trying to distinguish relationships between data variables.

Does background colour around the text make data easier to understand? (Figure 7.15 on page 116) Background colour was used minimally and evaluators required prompting about the option of putting background colour around the text. Although once used, most agreed it made the tag cloud easier



Q1 Are users aware of valid values for interface options?

Q2 Are measurement units displayed for data entry?

Q3 Is status information displayed?

Q4 Are labels provided for all fields?

Q5 Are cues provided on the acceptable length of data entries?

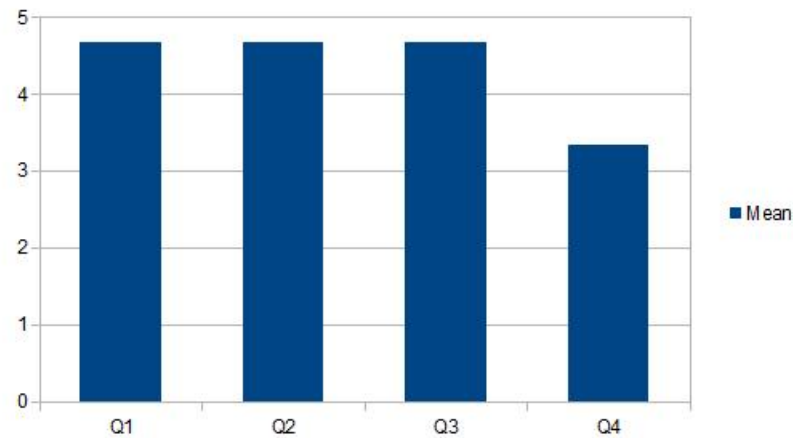
Q6 Are titles provided for each window?

Q7 Is online help and guidance provided?

Figure 7.12: *Prompting — interactivity*

to read.

Is a spiral or typewriter layout preferred? For what kinds of information? Evaluators did not establish a clear preference for spiral or typewriter layout for any particular dataset. The comment was made that “it really depended on the variables mapped”. One user felt that the spiral layout was useful for finding data with similar characteristics and that the typewriter layout affected user perception unnaturally, with arbitrary points for line breaking creating a disjoint in the data. However typewriter layout was perceived as useful for textual labels that benefited from being displayed in alphabetical order. There was an



Q1 I immediately understood that underlying data values between large tags and small tags were different

Q2 I immediately understood that underlying data values between tags with contrasting colours were different

Q3 I immediately understood that underlying data values between tags with contrasting transparency levels were different

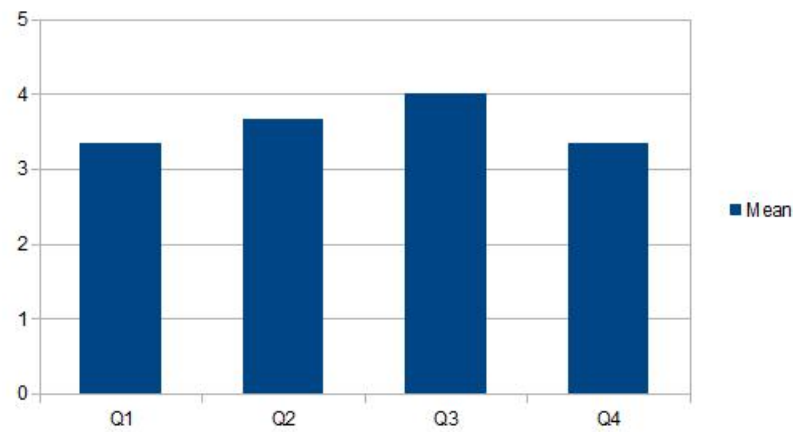
Q4 When looking at a visualisation the tag label gave me an immediate visual clue as to the kind of data the tag was representing

Figure 7.13: *Is the visualisation technique instinctively comprehensible?*

expectation by one evaluator that data with similar characteristics might cluster together or make exact circles in rows around the centre when using the spiral layout.

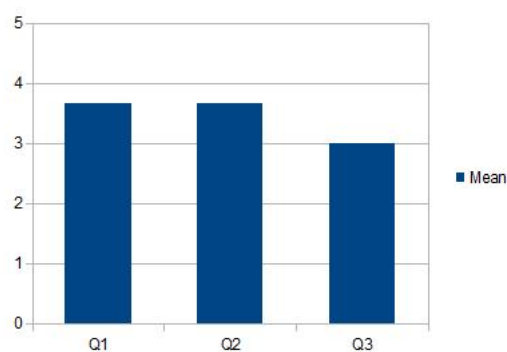
Is it useful to map a single data variable to multiple visual features?

(Figure 7.16 on page 117) All the users tried mapping a single data variable to multiple visual features. Users agreed it was useful to map a single data variable to multiple visual features. When mapping multiple data variables to multiple visual features the comment was made that the tag cloud “became less useful when mapping more than two data variables”.



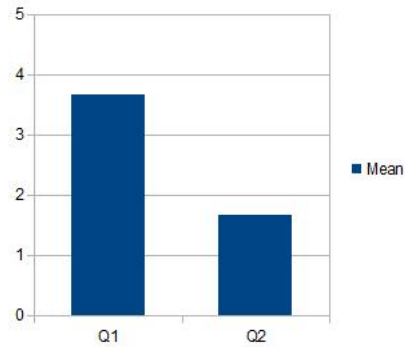
- Q1** I could infer the underlying data distribution of a dataset/datasets
- Q2** While interacting with the tag cloud, I could identify outliers of a dataset/datasets
- Q3** In a dataset/datasets, I could distinguish which tags had similar (data) characteristics
- Q4** I could distinguish a relationship between one data variable and another data variable

Figure 7.14: *Are users able to infer general information about the data from interacting with the visualisation? What kind of information?*



- Q1** Tag clouds were easier to read with background colour around the text
- Q2** Background colour around the text enhanced my knowledge of a particular dataset/s
- Q3** I forgot that I could put the colour around the background of the tag

Figure 7.15: *Does background colour around the text make data easier to understand?*



Q1 I found it useful to map a data variable to more than one tag cloud visual feature

Q2 I didn't try to map a data variable to more than one tag cloud visual feature

Figure 7.16: *Is it useful to map a single data variable to multiple visual features?*

7.7 Adaptations resulting from the evaluation

Having considered the evaluators' responses, we decided to modify Taggle to include some of their more significant suggestions.

Initial presentation of dataset Simpler default settings were selected. The background colour option was changed to be the default setting, as users agreed on its usefulness but forgot to use it. Linear mappings were set to be the default. Minimal visual features were mapped on cloud load — only tag text and order are automatically selected, with font size, colour and transparency mappings deselected. This is so not to overwhelm the user. Taggle was also altered to produce a cloud automatically after dataset selection to provide the user with a useful starting point.

Selection of data mappings Evaluators wanted the underlying data field names, ranges and data forms (such as text, number, category) more visible to the user to assist them with mapping selection. [Figure 7.17 on the next page](#) shows a scrollable, expandable data summary screen which was added above the tabbed panes showing information about each data field. This panel could be visible at all times while selecting mappings or applying filters, but could also be expanded or contracted to match user's preference or knowledge of the dataset.

For mapping selection data entry, editable fields applied rules to help the user

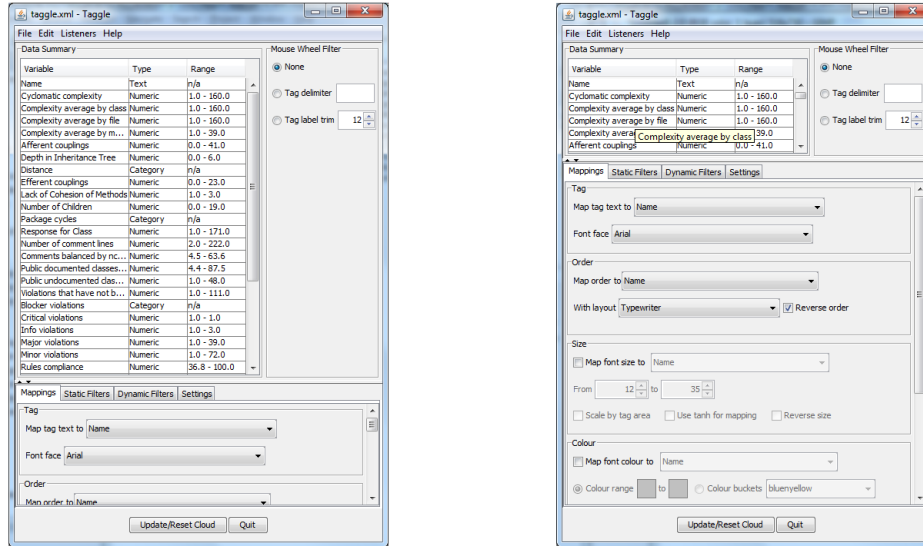


Figure 7.17: *Expandable data summary screen*

make more appropriate choices. For example, the font size maximum spinner buttons were changed to move the minimum font size to always less than the maximum number within the specified absolute minimum/maximum).

Mapping boxes were disabled when dechecked to make it absolutely clear that they were not being applied in the tag cloud. Layout algorithms were better labelled and unused items were disabled to minimise confusion (see Figure 7.18).

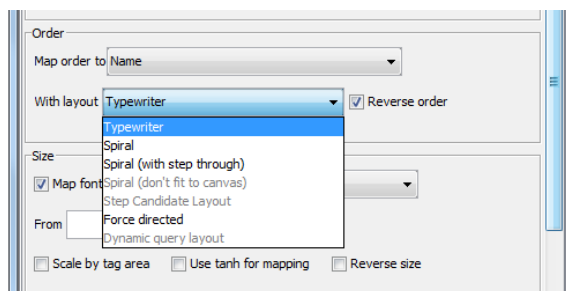


Figure 7.18: *Layout algorithm relabelled*

Evaluators used colour as a visual mapping property a lot, but were confused about which colour would be appropriate, and tended to use the default settings. It was also noted that the colour mapping was best used for continuous data rather than categorical; colours were presented on a scale with a colour gradient

between a minimum and maximum value range. Figure 7.19 shows a change providing an optimal option for categorical data — colour buckets, where each colour represented a category value. If the mapped data field is categorical then the colour bucket option is automatically selected, otherwise the colour range option is used. For colour buckets, there is a list of colour sets available to use rather than total flexibility in colour choice by user. Colour schemes with appropriate contrast can be entered into an external property file and this will be loaded into the colour bucket drop-down box on Taggle initiation.

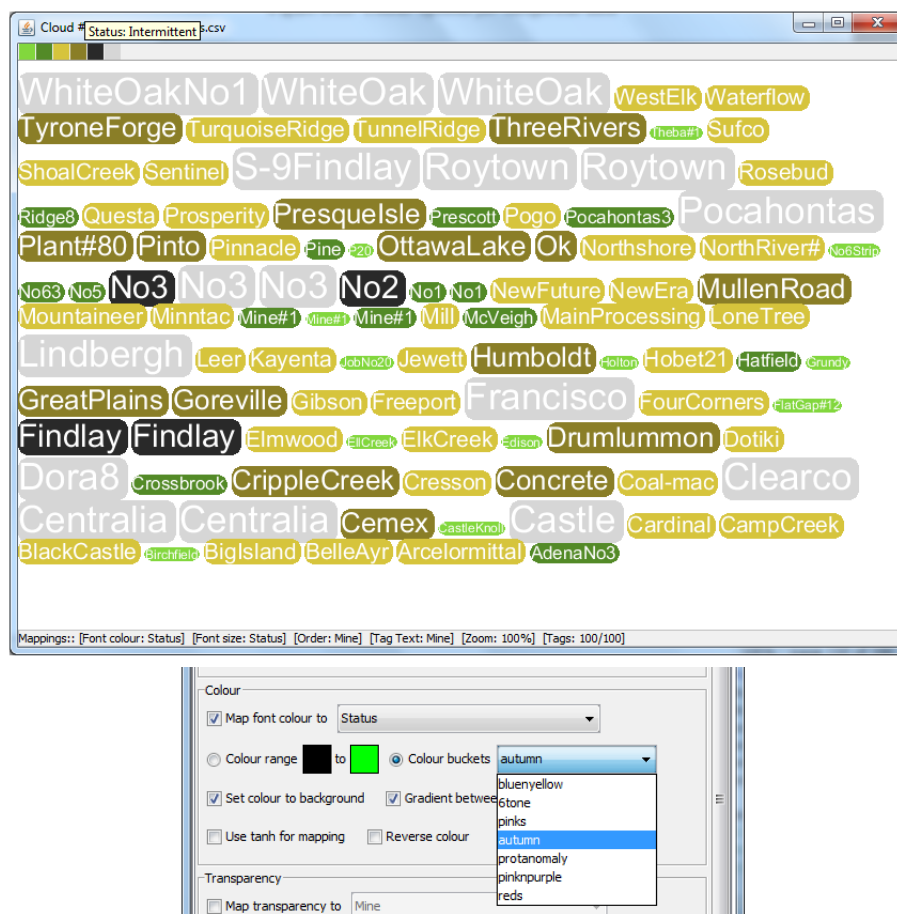


Figure 7.19: *Colour options for categorical data — brown tags are US mines with an operating status of ‘Intermittent’*

Static and dynamic filtering The mouse wheel filtering of the tag text was perceived to be the most important filtering option and evaluators requested that the options be more visible to users, especially when performing mapping variable selection. Therefore, these options were moved to the new expandable data summary panel to be viewed at all times (see Figure 7.17 on page 118). A third filtering option “None” was added, and set to be the default. The maximum label size was renamed a more user-friendly ‘Tag label trim’.

Visual encoding One aspect of the visual encoding which evaluators wished included was to have categorical data complemented with an associated colour box in the legend. This has been implemented with a fix to the mapping selection colour property (see Figure 7.19 on the preceding page).

The overview pop-up in the visual encoding was changed to display on hovering over a tag rather than on right-click, so that it was easier to find.

7.8 Summary and discussion

We felt that through the combination of heuristic issue identification, heuristic guideline checklist, questionnaire and general observation and comment gathering, that we had a reasonable idea of the answers to our initial research questions.

RQ: *Is the visualisation technique itself instinctively comprehensible? Can the visualisation supply the user with a good overall picture of the data?* Users felt they instinctively comprehended that tags with contrasting font sizes, colours or transparencies had differing underlying data values. The tag itself didn’t always give an immediate visual clue to represented data as it relied on appropriate user selected mappings.

RQ: *Can users infer general information about the data from interacting with the visualisation? What kind of information?* Users were able to infer underlying data distributions, identify outliers and distinguish tags with similar characteristics. This again relied on useful user selected mappings, so difficulty levels varied among evaluators.

RQ: *Is the data mapping process understandable?* Overall, this was the area which evaluators had the most trouble with. Evaluators felt they needed to be supplied with more information about the dataset to support them in choosing appropriate data mappings, since selection of mappings was the most important step in the creation of a cloud which would enable them to refine their knowledge of the underlying data.

RQ: *Is the supplied system interactivity sufficient for accumulating knowledge about the data?* There were a number of suggestions made during the heuristic evaluation about how to improve the interactivity of the system. These suggestions were generally around the area of mapping selection. Filtering features were not used extensively in the evaluation — this could have been improved by providing the evaluators with more specific questions about the given datasets.

Two of our posed questions in the questionnaire were relating to tag background colour and dual mappings, this served as a initial sanity check for our two planned visual encoding controlled experiments. Tag background colour was agreed to be useful especially when mapping multi-word textual data to the tag label, but as it wasn't set on by default, it wasn't applied until evaluators were reminded that it was an option. Dual mappings for one data field were also perceived as useful when asked. It was commented that multiple mappings for multiple variables became unhelpful when mapping more than two data fields. Based on this primer, we felt confident there were no serious issues in continuing with the planned experiments (presented in Chapters 8, 9, and 10). As a result of the comments observed and discussed in the heuristic evaluation, a number of adaptations were made to the Taggle prototype in order to improve usability.

8

Experiment One: Tag Colour Placement

A series of experiments was conducted to explore the potentials and limitations of the interactive tag cloud visualisation tool Taggle. Properties of the enhanced tag cloud features utilised in the system were investigated in two experiments using eye-tracking technology to discover if improvements could be made in user performance for visual search tasks. This chapter details the first experiment, where we investigated manipulation of the tag colour placement in the font and tag background. Research questions and goals for the study are presented in §8.1. The methodology of the evaluation is detailed from §8.2 through §8.9. Results from the response times and the eye-gaze data analysis are discussed in §8.10, with threats to the experiment validity considered in §8.11. Finally, results are summarised and conclusions are presented in §8.12 and §8.13.

8.1 Foreground or background colour

A special feature of the tag cloud visual encoding in Taggle is the optional use of colour around the tag background instead of the font. This has the benefit of increasing the amount of colour which the user needs to locate in a visual search. Larger areas are better for distinguishing colours (see §3.1.7 for more details). From a software engineering visualisation perspective, this is important as users of our tag cloud tool will need to deal with correlations between data variables mapped to size and colour. Large datasets, such as those found in software engineering, will inevitably contain small tags. Realising distinguishing colour becomes harder when the tag is smaller, we were interested to find out if increasing the area of colour in a tag could provide a search performance time improvement.

RQ: How does altering tag colour placement between font or tag background affect user performance for search tasks?

Further to this, we wanted to find out if a change in layout or tag size made a difference to any performance change.

Five tag cloud sets consisting of eight tag clouds containing 100 tags each were created within a canvas of 850×650 pixels (40 tag clouds). The canvas size was determined by screen real estate constraints, while allowing enough room to see layout differences and providing for an average tag length of eight characters across the datasets. Participants saw 32 tag clouds each, plus a trial set of eight tag clouds.

Each set of tag clouds contained a different colour palette of six colours to minimise the effect of colour combinations having a different impact on visual perception between individual participants. Each colour palette contained a set of six contrasting colours so that we could be reasonably assured participants could distinguish each colour. Order and size mappings within each tag cloud were varied according to a randomly generated variable. Font sizes were set between 15 pt and 45 pt.

8.2 Participants

We recruited ten University of Canterbury students to participate in the study (age 18–35, all male, with no reported uncorrected vision problems). All had completed, or were completing stage two or three computer science university papers. Participants received a \$20 gift voucher for participating in the study.

8.3 Apparatus

Tag clouds were presented on a 17-inch LCD screen with 1280×1024 pixels. The study was run utilising a Tobii T60 eye-tracking machine attached to the monitor. Eye-gaze data was collected with Tobii Studio 3.2.1 software using a minimum fixation duration of 60 milliseconds. Eye movements of each participant were calibrated to five points before beginning the tasks.

8.4 Tag corpora

Five tag corpora were developed containing 100 data points each, with a categorical variable containing six categories. These were created from knowledge areas we expected were lesser known to our subjects to minimise bias. Textual data (Table 8.1 on the next page) was sourced from real-life datasets retrieved from a variety of sources such as the IUCN list of endangered species¹, Find The Data² and the Parliament of Australia³.

8.5 Procedure

After completing a demographic questionnaire and signing a consent form, five minutes of training describing tag cloud visualisation was given pre-experiment.

Participants were shown sets of tag clouds varying colour placement, size, and layout, with order counter-balanced using a Latin square design. Each set of tag

¹<http://www.iucnredlist.org/>

²<http://www.findthedata.org/>

³<http://www.aph.gov.au/>

Table 8.1: *Datasets used in repetitions*

Repetitions	Dataset	Categorical variable
Training	Cross country team members	Team name
No. 1	Australian MPs	Political party name
No. 2	NZ birds	Level of endangerment
No. 3	US counties	US state
No. 4	US mines	Mining status

clouds related to a particular tag corpus, where the target tag was identical for each cloud. Before each set of clouds was displayed, a short text was displayed explaining the scenario and naming a target tag. This text was the same for each cloud set with the scenario adapted to the corresponding tag corpus. Target tags were selected randomly, with the number of data points within the target category varying between corpus.

The experiment began with a practice tag cloud set containing eight tag clouds. Following this, four tag cloud sets containing eight tag clouds each (timed trials) were presented (a total of 32 timed trials per participant).

To minimise time and bias, participants performed experiment one and three together, but not experiment two. Overall, participation took 40 to 60 minutes.

8.6 Design

We used a $2 \times 2 \times 2$ experimental design for the following within-subject factors and levels:

- Colour placement {background, foreground}
- Size of tag {smaller, larger}
- Layout {spiral, typewriter}



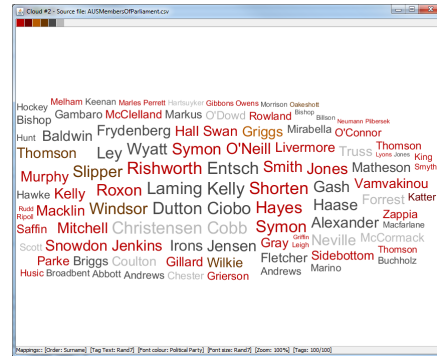
(a) Background colour, typewriter layout



(b) Background colour, spiral layout



(c) Font colour, typewriter layout



(d) Font colour, spiral layout

Figure 8.1: *Colour and layout in experiment one: You are about to see names of Australian members of parliament from different political parties. Red names belong to the political party 'Australian Labour Party'. Find and click on 'Murphy' from the Australian Labour Party.*

8.7 Task

Participants completed a visual search task with a random target belonging to a particular category in a tag cloud containing 100 tags. The target was specifically named and there was only one such target within each tag cloud. The number of tags in the target category were distributed across the repetitions (9, 12, 27 and 48). The scenario for the dataset and specific target was described in the instruction screen before the trial, as shown in Figure 8.1.

8.8 Measurements

The dependent measure was response time (time to first mouse click on target). This was captured by the presentation software between the media first appearing and the user clicking the mouse button on a target. Eye-gaze data was also collected. There was only one target tag in each tag cloud and participants had as much time as they needed to locate the target. There were no incorrect targets selected in the experiment.

8.9 Hypotheses

We defined two hypotheses:

H1: In general, the larger the area of colour in a colour-coded field, the more easily it can be distinguished [pg 125, ch 4 Ware, 2004]. Therefore, we hypothesize that *the greater area of colour present in a tag background will lead to faster visual search times for tag background colour than tag foreground colour.*

H2: *A performance increase for visual search time using tag background colour will be greater for smaller tags than larger tags.*

8.10 Results

8.10.1 Eye-gaze data analysis

The analysis of the eye-tracking data was performed manually, in an exploratory manner looking for typical patterns in the visual search. Previous research Schrammel et al. [2009a] analysing scan paths in tag clouds has identified two typical search patterns within tag clouds; chaotic scanning (no traceable strategy) and serial scanning (following a zig-zag path). Our exploratory analysis on collected data indicates the introduction of a colour hint in tag searches can alter the search strategy, with the eye scan path focusing on tags with the target

Table 8.2: *Search strategies in tag clouds*

Search Type	Gaze Path Description
Serial scanning	Scanning left to right or right to left successively
Feature search	Fixation clustering around tags with a specific visual feature (colour)
Chaotic search	No fixed pattern

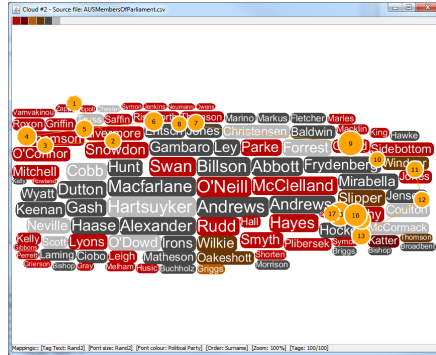
colour. Table 8.2 on the next page describes the search strategies found in tag clouds in experiment one.

Visual feature search was found in all repetitions (which featured different datasets and a different colour palette — see Figure 8.2 on the following page). However, in datasets containing a larger target category size such as in repetition one (48 tags), some participants used chaotic or serial scanning searches to locate the target. This was also the case in repetition three which had a medium-sized target category (27 tags), but gaze analysis showed participants became confused between the target colour and another closely related colour (light purple and dark purple), increasing the target category size to 47. Figure 8.3 on page 130 shows examples where in trial repetitions containing datasets with a larger target category, participants also employed combination searches switching between multiple search strategies.

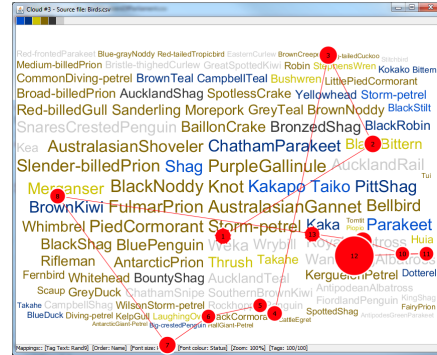
8.10.2 Response time

Figure 8.4 on page 130 shows response time across colour placement, size and layout and reveals something unexpected in the spiral layout data. While all other colour/layout groups show large tags with a performance advantage over small tags, group foreground colour and spiral layout (FG/SP) show small tags being located faster than large tags. Because of this unexpected and seemingly counter-intuitive result (coupled with the large standard error for that group) the data was carefully screened for the presence of outliers and possible bias.

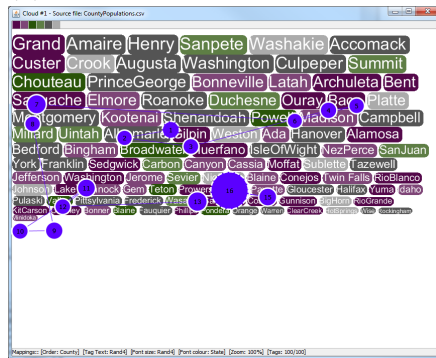
Data was checked for possible bias in target location between the groups. Target location was assigned randomly to a stimulus. Previous research analysing



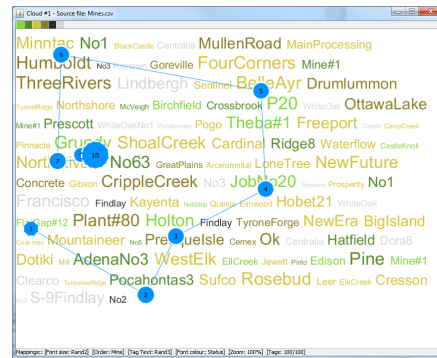
(a) Repetition one with red target



(b) Repetition two with yellow target



(c) Repetition three with purple target



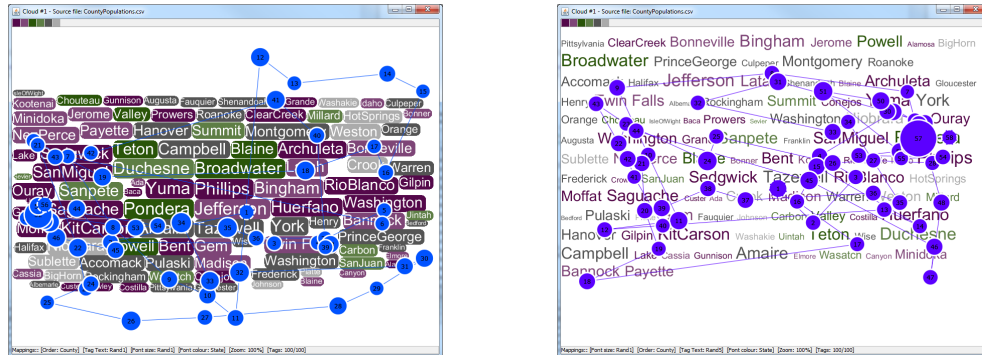
(d) Repetition four with green target

Figure 8.2: *Visual feature search with fixation clustering around tags with target colour. Animated visualisations of these examples can be found at <http://www.cosc.canterbury.ac.nz/research/RG/svg/taggle/>*

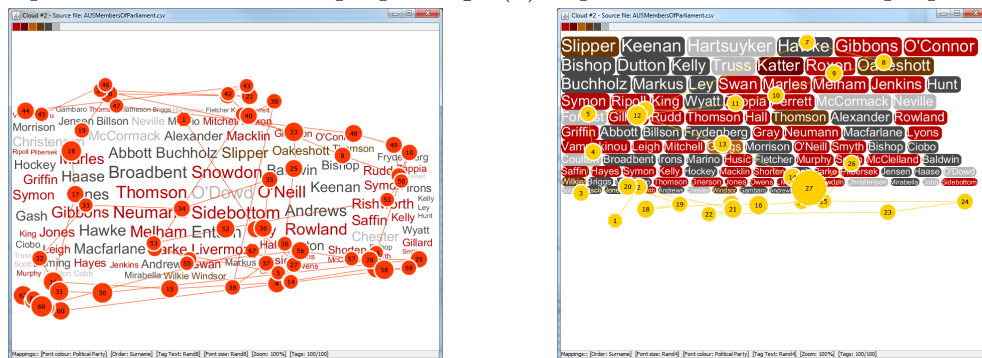
eye-gaze data for tag clouds has determined initial gaze locations tend towards a central distribution [Lohmann et al., 2009]. It is possible that if the target is positioned further away from a central point that it is found less quickly. We checked to see if randomly placed target positions had resulted in more unfavourable locations for some experimental conditions.

The average location of the first eye fixation across all conditions was calculated. This was close to the stimulus midpoint, but inside the upper left quartile — see Figure 8.5 on page 131.

The distance in pixels between the target tag and average location of first fixation for all stimuli was calculated (4 repetitions \times 8 conditions = 32 visual stimuli). A pictorial representation of a subset of these can be seen in Figure 8.6 on page 131 where a) shows a favourably located target and b) shows a mildly



(a) Repetition three with dark purple target (b) Repetition three with dark purple target



(c) Repetition one with red target (d) Repetition one with red target

Figure 8.3: *Visual searches with combinations of chaotic search, serial scanning and fixation clustering around tags with target colour. Animated visualisations of these examples can be found at <http://www.cosc.canterbury.ac.nz/research/RG/svg/taggle/>*

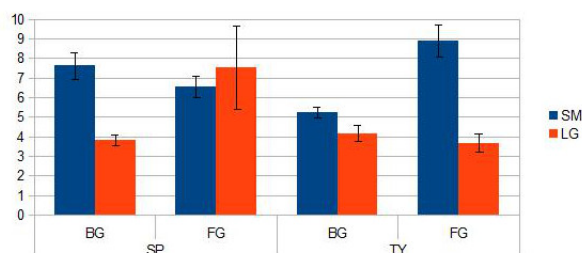


Figure 8.4: *Visual search response time*

unfavourably located target (representations of all 32 visual stimuli can be found in Appendix C). Average distance to first fixation (Figure 8.7 on page 132) can be compared with response time results (Figure 8.4) . Response times and average

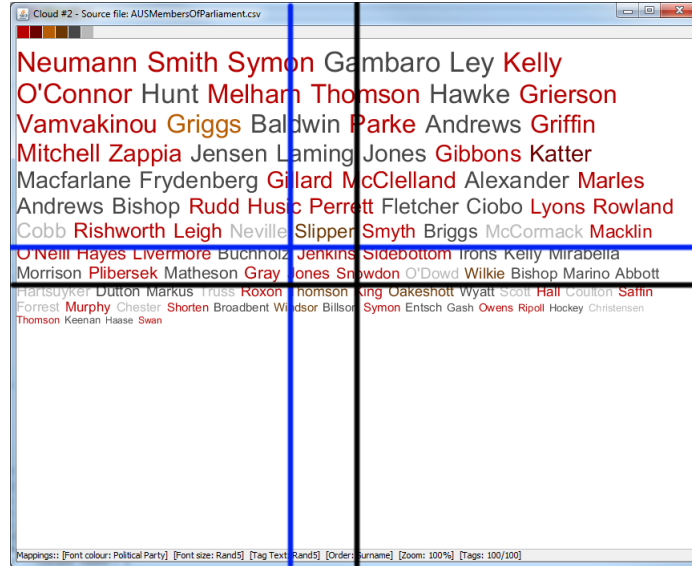


Figure 8.5: Average location of first fixation (blue) and stimulus midpoint (black)

location don't match in the FG/SP condition. Although large tags are more favourably located than small tags, the small tags were found more quickly.

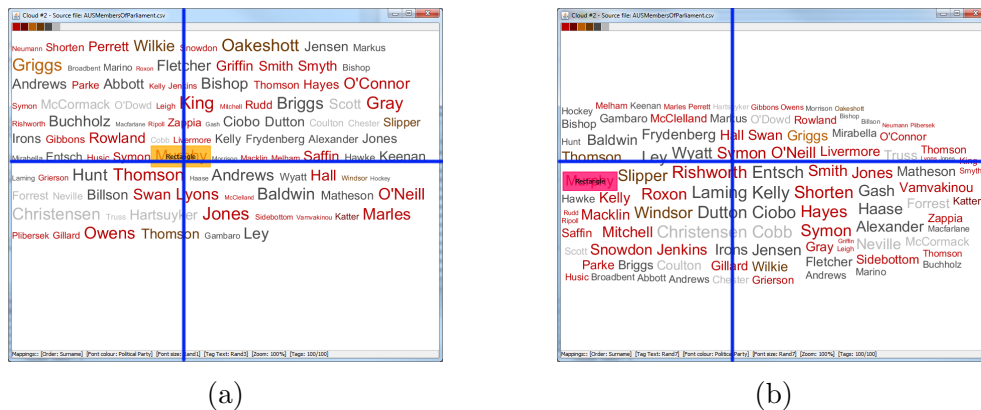


Figure 8.6: Foreground colour with large target, typewriter and spiral layout

For each condition in each repetition one of four categories for location was assigned; Favourable, Mildly Favourable, Mildly Unfavourable, Unfavourable. Assignment to location category was based on whether the distance was above/below median or 1st and 3rd quartile fixation values (see Table 8.3 on the next page).

A large number of outliers (42) were calculated outside of three standard deviations from the condition mean. Outliers did not appear to be greatly related

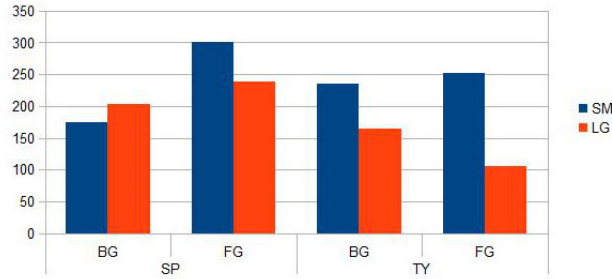


Figure 8.7: *Average distance in pixels to first fixation*

Table 8.3: *Target location favourableness per condition and repetition*

Stimulus	R1	R2	R3	R4
BG/SM/SP	Mildly favourable	Mildly favourable	Mildly unfavourable	Mildly favourable
BG/SM/TY	Favourable	Unfavourable	Favourable	Unfavourable
BG/LG/SP	Unfavourable	Unfavourable	Favourable	Favourable
BG/LG/TY	Mildly unfavourable	Favourable	Mildly favourable	Mildly favourable
FG/SM/SP	Unfavourable	Unfavourable	Mildly unfavourable	Mildly unfavourable
FG/SM/TY	Mildly unfavourable	Favourable	Unfavourable	Unfavourable
FG/LG/SP	Mildly unfavourable	Mildly favourable	Mildly unfavourable	Mildly unfavourable
FG/LG/TY	Favourable	Mildly unfavourable	Favourable	Mildly favourable

to presentation in repetition groups with mildly unfavourable or unfavourable target locations (59 percent). The greatest factor correlated to the presence of outliers was target category size. Outliers were mostly contained to repetition one and repetition three (90 percent) which used datasets containing categories with a large and medium number of data points (48 and 27). Outliers were spread fairly evenly across all eight conditions, and spread across all but one participant. Data distribution is positively skewed (see Figure 8.8 on the following page). This positive skew is a characteristic of response times in general [Luce, 1986; Zandt, 2000] and visual search response time [Palmer et al., 2011].

Eye-gaze data for all outlying data points was also analysed. Analysis of

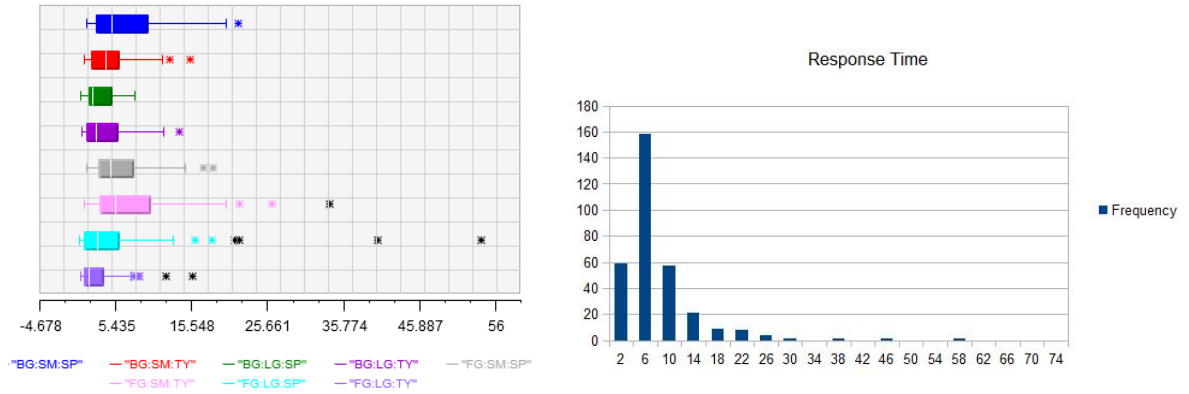


Figure 8.8: *Distribution of data*

eye-gaze data concluded participants often used the colour visual feature to aid their search for the target tag (§8.10.1). However, when presented with a larger number of coloured targets such as in trial repetition one, participants appeared to resort to using a random search method or serial scanning to locate the target. It appeared that in repetition three, participants were confused by the presence of two colours (light and dark purple) which were arguably less strongly contrasting than other pairs. Together these two colour made a large proportion of the dataset (47 tags). Participants searched for the target by either scanning both colours or resorting to using random/serial scanning methods.

Analysis of the response time data revealed the presence of a number of outliers and strongly positively skewed data. Careful examination of eye-tracking data showed outliers were due mostly to a variance in category size which increased the complexity of the task for the participant, rather than experiment procedure or participant error.

8.10.3 Significance testing

A two-way ANOVA with repeated measures on the response times data showed a significant main effect for factor size ($F_{1,39} = 52.5$, $p < 0.000001$), and a three-way interaction between colour placement, tag size and layout ($F_{1,39} = 7.13$, $p < 0.011008$). A simple two-way interaction between colour placement and font size within the typewriter layout was revealed during post-hoc analysis.

Bonferroni-corrected pairwise comparisons showed tag colour placement in the background produced significantly faster visual search response times than tag cloud foreground placement when the target tag size was small ($t(39) = 2.90$, $p < 0.0061$). This effect was achieved only when using the typewriter layout. Differences in search response time for large tags with background colour placement and foreground colour placement were not statistically significant in either spiral or typewriter layout. Visual search time for large tags was faster than small tags with an average response time of $t = 4.79s$ compared to $t = 7.06s$ ($t(159) = 3.58$, $p < 0.0005$). All statistical testing was performed applying a logarithm transform so ANOVA data normality assumptions could be met. See Figure 8.9 for normalised response times and interactions for typewriter and spiral layout data.

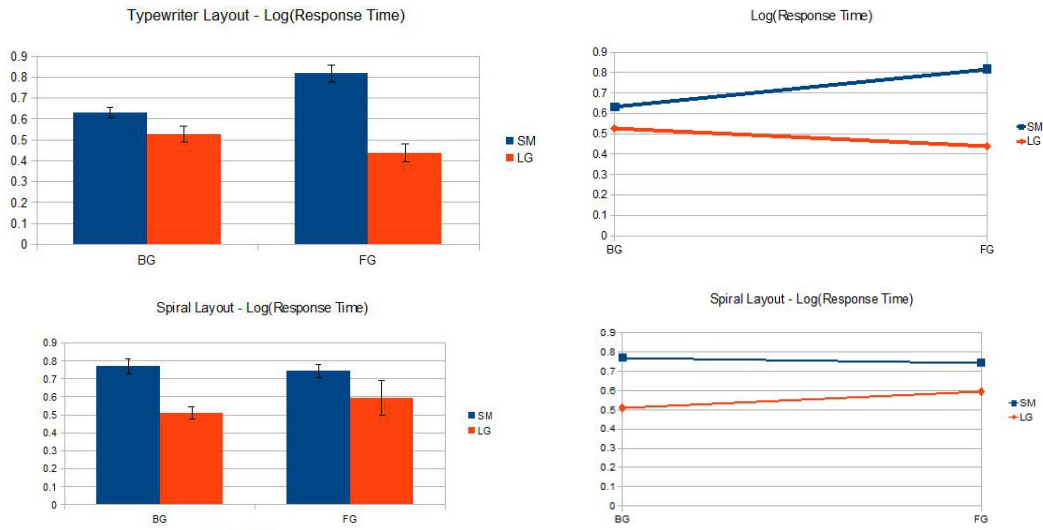


Figure 8.9: *Normalised response times and interactions*

8.11 Threats to validity

Usage of genuine categorical datasets, while realistic, can add challenges. It is possible that the use of real text with long words increasing text size has introduced bias, as this has been shown to have an effect on user perception of tag importance [Bateman et al., 2008]. However, this may be mitigated by the

visual search task type as users were not identifying tags of subjective importance, but searching for a specifically named tag.

Some of the experiment conditions have (averaged across across all stimuli shown) an overall more potentially favourable target location than other conditions (it is unknown how much influence distance to average first fixation point has on search time). This does not appear to have resulted in a pattern in the response times or impacted the conditions where outliers have appeared.

8.12 Summary and discussion

Analysis of the response time data revealed an unexpected result in foreground colour and spiral layout conditions — small tags were found more quickly than large tags. This is counter-intuitive and also contravening results in previous studies on the effect of font size on user perception in tag clouds (see §3.1.4 for details of this). Eye-tracking data did not show a bias in target location that might explain this result, as large tags were more favourably placed than small tags for this condition. Careful analysis of outlying data points indicated outlier presence seemed related to target category size, and overall distribution showed a significant positive skew. Therefore, all statistical testing was completed using a logarithm transform so data would conform to statistical testing requirements of normality.

H1: *Placement of colour in the tag background produced faster visual search times than tag foreground colour, but only when searching for small tags, and only when the tag cloud was displayed in a typewriter layout.* Placement of tag colour made no difference to response time in a tag cloud displayed in a spiral layout. We don't know why the results were different for layouts: whether there is some inherent difference in the way users search for visual targets in a spiral layout, or there was something particular to the spiral layout algorithm used in our experiment. Further close analysis of the eye-tracking data could be performed in the future to explore this further.

H2: *The performance increase seen in typewriter layout for visual search time using tag background colour was only when locating smaller tags and NOT larger tags.* As expected, using a greater area of colour to search for when locating tags

was of benefit mostly when searching for tags with smaller sized fonts.

Analysis of eye-tracking data indicates the introduction of a colour hint in tag searches can alter the search strategy from serial scanning or chaotic search methods, to the eye scan path focusing on tags with the target colour. In trial repetitions containing datasets with a larger target category, participants also employed combination searches switching between multiple search strategies.

8.13 Conclusion and future work

Our results indicate usage of tag background colour as a data variable field can produce faster visual search response times than foreground colour when the target tag is small. This has implications for designers of tag cloud visualisations. Small tag sizes are inevitable in tag clouds, especially when visualising large datasets. We think providing the background colour option as the default setting in our interactive tag cloud visualisation tool could help users in the following ways: identifying the variables mapped to colour (particularly those displayed with small or iconified tags), and exploring correlations more efficiently between data variables mapped to size and colour.

This experiment has provided some insight into tag cloud visual search patterns. Varying target category size over repetitions produced some abnormally long response times (outliers) for categories with greater sizes. Eye-tracking data analysis showed when searching for targets in larger sized categories, participants employed combination search methods switching between visual feature search (with colour), serial scanning and chaotic search.

For future experiments it would be interesting to focus on category size as an experiment factor to further explore eye scanning methods and to discover at what point (such as an overall percentage or specific tag number) participants start producing greater numbers of atypical long response times. Our experiment produced differing results between the two layouts tested, spiral and typewriter. The reasons for this are unclear. Future eye-tracking data analysis or experiments may focus on finding out more about this difference.

9

Experiment Two: Dual Mappings

A series of experiments was conducted to explore the potentials and limitations of the interactive tag cloud visualisation tool Taggle. Properties of the enhanced tag cloud features utilised in the system were investigated in two experiments using eye-tracking technology to discover if improvements could be made in user performance for visual search tasks. This chapter details the second experiment, where we investigated dual data mappings to font size and colour visual properties. Research questions and goals for the study are presented in §9.1. The methodology of the evaluation is detailed from §9.2 through §9.9. Results from the response times and the eye-gaze data analysis are discussed in §9.10, with threats to the experiment validity considered in §9.11. Finally, results are summarised and conclusions are presented in §9.12 and §9.13.

9.1 Single or dual data mappings

A special feature of the tag cloud visual encoding in Taggle is that we can use multiple visual features for visualising data variables. Previous research [Bateman et al., 2008; Halvey and Keane, 2007; Lohmann et al., 2009] shows that different visual features have different effects on user perception when searching for a particular tag. From a software engineering visualisation perspective, this is important as users of our tag cloud tool will need to deal with correlations between data variables mapped to multiple visual features. We wanted to find out what effect mapping dual visual properties had — if using two visual features such as size and colour together to represent a data field would have an improved user visual search performance than one visual feature. We had the following research question:

RQ: How does mapping a data variable to size or colour compare to mapping a data variable to size and colour together with respect to user performance for search tasks?

We also wanted to find out if a change in layout made a difference to performance. Five tag cloud sets consisting of six tag clouds containing 100 tags each were created within a canvas of 850×650 pixels (30 tag clouds). Participants saw 24 tag clouds each, plus a trial set of six tag clouds. For details regarding how the canvas size was determined and for tag cloud construction information, see §8.1.

9.2 Participants

We recruited ten University of Canterbury students to participate in the study (age 18–35, 1 female, with no reported uncorrected vision problems). All had completed, or were completing stage two or three computer science university papers. Participants received a \$20 gift voucher for participating in the study.

9.3 Apparatus

For details regarding the apparatus used, see §8.3.

9.4 Tag corpora

Five tag corpora were developed containing 100 data points each, with a categorical variable containing six categories — refer to §8.4 for details.

9.5 Procedure

After completing a demographic questionnaire and signing a consent form, five minutes of training describing tag cloud visualisation was given pre-experiment.

Participants were shown sets of tag clouds varying data to visual property mappings, and layout, with order counter-balanced using a Latin square design. Each set of tag clouds related to a particular tag corpus, where the target tag was identical for each cloud. Before each tag cloud, a short text was displayed explaining the scenario and naming a target tag. This text was the same for each cloud set with the scenario adapted to the corresponding tag corpus, and the corresponding visual property hint given. Target tags were selected randomly, with the number of data points within the target category varying between corpus.

The experiment began with a practice tag cloud set containing six tag clouds. Following this, four tag cloud sets containing six tag clouds each (timed trials) were presented (a total of 24 timed trials per participant).

To minimise time and bias, participants performed experiment two and three together, but not experiment one. Overall, participation took 40 to 60 minutes.

9.6 Design

We used a 3×2 experimental design for the following within-subject factors and levels:

- Mappings {size, colour, dual}

-
- Layout {spiral, typewriter}

The dual level in the mappings factor refers to size and colour visual properties being used together to map the same data variable.

9.7 Task

Participants completed a visual search task with a random target belonging to a particular category in a tag cloud containing 100 tags. The target was specifically named and there was only one such target within each tag cloud. The number of tags in the target category was distributed across the repetitions (9, 16, 21, and 48). The scenario for the dataset and specific target was described in the instruction screen before the trial, as shown in [Figure 9.1 on the following page](#).

9.8 Measurements

The dependent measure was response time (time to first mouse click on target) – this is the same as in experiment one, see §8.8 for details.

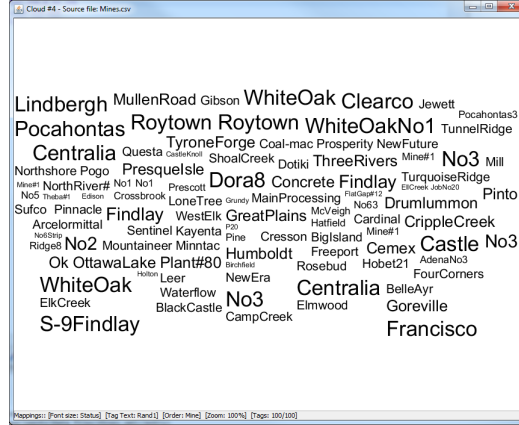
There were no incorrect targets selected in the experiment. In one trial, a participant could not recall the name of the target. This trial was not included in the statistical analysis.

9.9 Hypotheses

We defined a hypothesis:

H1: *Mapping a data field to tag font size and colour will lead to faster visual search times than mapping a data field to either tag font size or colour separately.*

*You are about to see
names of mines in the
US and their operating
status. Names with a
large font size have the
status ‘Non-producing’.
Find and click on
‘Dora8’ which has
status non-producing.*



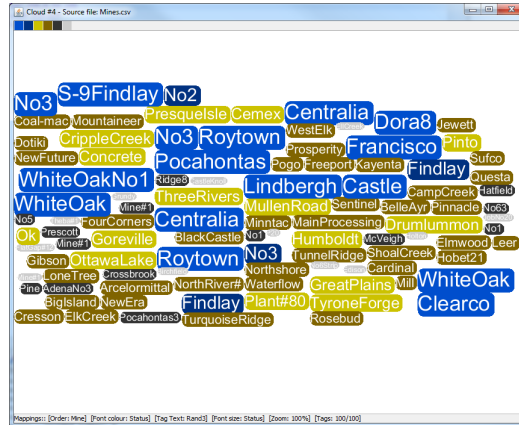
(a) Font size mapping, spiral layout

*You are about to see
names of mines in the
US and their operating
status. Light blue
colour names have the
status ‘Non-producing’.
Find and click on
‘Dora8’ which has
status non-producing.*



(b) Tag colour mapping, typewriter layout

*You are about to see
names of mines in the
US and their operating
status. Light blue
colour names with a
large font size have the
status ‘Non-producing’.
Find and click on
‘Dora8’ which has
status non-producing.*



(c) Tag colour + font size mapping, spiral layout

Figure 9.1: Mapping and layout combinations in experiment two

9.10 Results

9.10.1 Eye-gaze data analysis

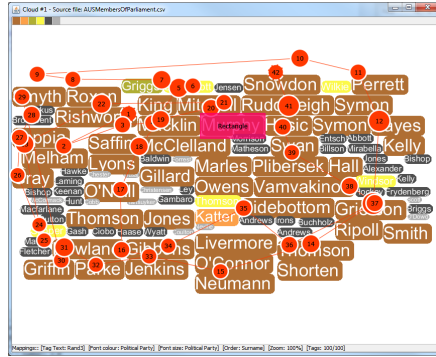
The analysis of the eye-tracking data was performed manually, in an exploratory manner looking for typical patterns in the visual search. As with the eye-gaze data analysis from experiment one, collected data indicates the introduction of a colour hint in tag searches can alter the search strategy, with the eye scan path focusing on tags with the target colour. Refer to §8.10.1 for details describing the search strategies found in tag clouds in experiment one and experiment two.

Visual feature search was found across all three mapping types and all repetitions (which featured different datasets and a different colour palette where applicable — see Figure 9.2 on the next page). As in experiment one, with datasets containing a larger target category size (such as repetition one — 48 tags), some participants used chaotic or serial scanning searches to locate the target.

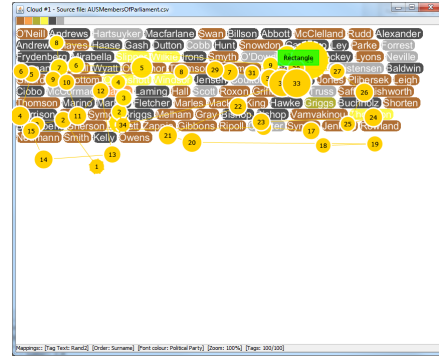
In repetition two, the target category size was much smaller (21 tags), but gaze analysis showed the tag cloud created for the underlying dataset took up more space proportionally on the canvas than tag clouds from other repetitions (see §9.10.2 for further discussion regarding this). With more text to search through (and a longer response time overall), participants appeared to be more likely to switch between search methods. Figure 9.3 on page 144 shows examples from trial repetitions one and two, where participants employed combination searches switching between multiple search strategies.

9.10.2 Response time

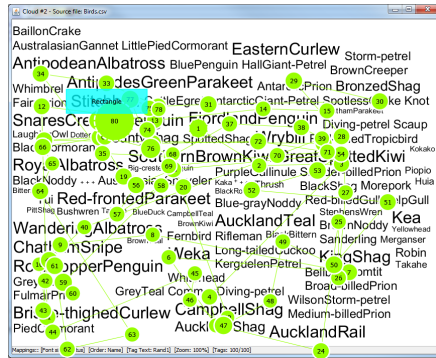
Figure 9.4 on page 144 shows visual search response time across mappings (size, colour and dual) and layout type. While time differences in spiral layout don't appear to vary much between singular and dual mappings, there is a difference across the typewriter layout. Analysis of the tag cloud set-up in experiment one detailed in Chapter 8 identified a possible bias in target location between the conditions. Therefore, data was also checked for bias in target location in experiment two.



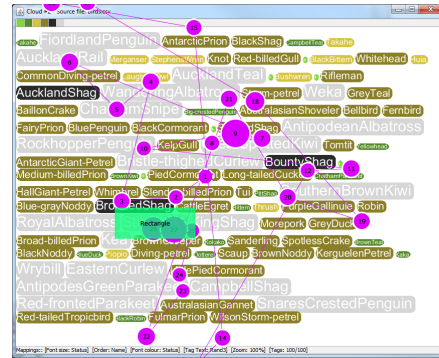
(a) Repetition one with dual mapping



(b) Repetition one with colour mapping



(c) Repetition two with size mapping



(d) Repetition two with dual mapping

Figure 9.3: Visual searches with combinations of chaotic search, serial scanning and fixation clustering around tags with target mapping. Animated visualisations of these examples can be found at <http://www.cosc.canterbury.ac.nz/research/RG/svg/taggle/>

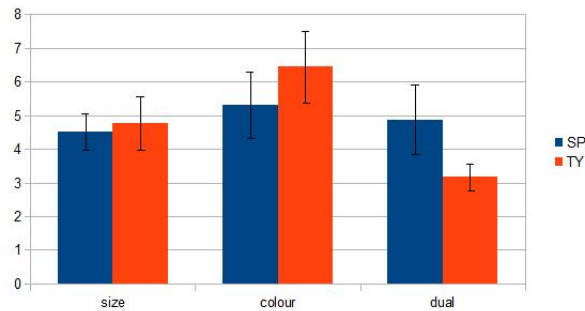


Figure 9.4: Visual search response time in seconds

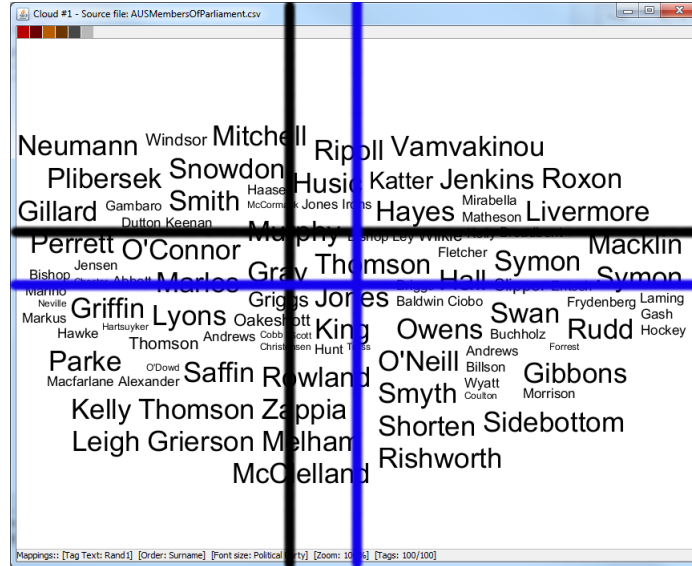


Figure 9.5: Average location of first fixation (black) and stimulus midpoint (blue)

in Appendix C). Average distance to first fixation (Figure 9.7 on the following page) can be compared with response time results (Figure 9.4 on the previous page). There is an average difference of 210 pixels to the first fixation point between spiral and typewriter layouts for the singular font size mapping. Despite this seemingly large difference, visual search average response times for singular font size mappings between spiral and typewriter layout are very similar ($t = 4.51s$ and $t = 4.77s$). It is unclear how much influence distance to average first fixation point has on search time.

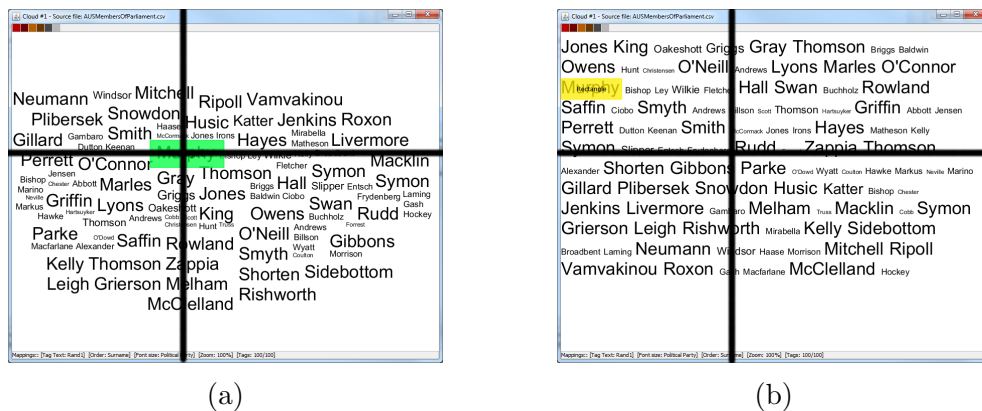


Figure 9.6: Singular font size mapping, spiral and typewriter layouts

Table 9.1: *Target location favourableness per condition and repetition*

Stimulus	R1	R2	R3	R4
size/spiral	Favourable	Favourable	Unfavourable	Favourable
size/typewriter	Mildly unfavourable	Mildly unfavourable	Favourable	Unfavourable
colour/spiral	Mildly favourable	Unfavourable	Mildly favourable	Mildly favourable
colour/typewriter	Mildly unfavourable	Mildly unfavourable	Unfavourable	Unfavourable
dual/spiral	Favourable	Unfavourable	Mildly favourable	Mildly unfavourable
dual/typewriter	Favourable	Favourable	Mildly favourable	Mildly favourable

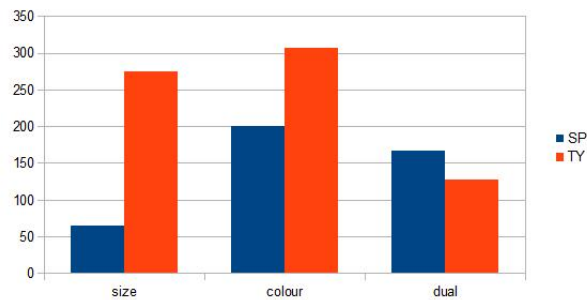


Figure 9.7: *Average distance in pixels to first fixation*

For each condition in a repetition one of four categories for location was assigned; Favourable, Mildly Favourable, Mildly Unfavourable, Unfavourable. Assignment to location category was based on whether the distance was above/below median or 1st and 3rd quartile fixation values (see Table 9.1).

Several outliers (seventeen) were calculated outside of three standard deviations from the condition mean. Only 29 percent of outliers were in groups with mildly unfavourable or unfavourable target locations, so outlier presence was not related to a poorer level of target location favourableness. The greatest factor correlated to the presence of outliers was repetition/dataset. Outliers were mostly contained to repetition one and repetition two (88 percent). The greatest number of outliers (11) were found in repetition two which had 21 data points in the tar-

get category, the next largest target category contained 16 data points and had no outliers. Outliers were spread fairly evenly across all eight conditions, and spread across all but one participant. Data distribution is positively skewed (see Figure 9.8) — skewness is characteristic of response times [Luce, 1986; Palmer et al., 2011; Zandt, 2000]).

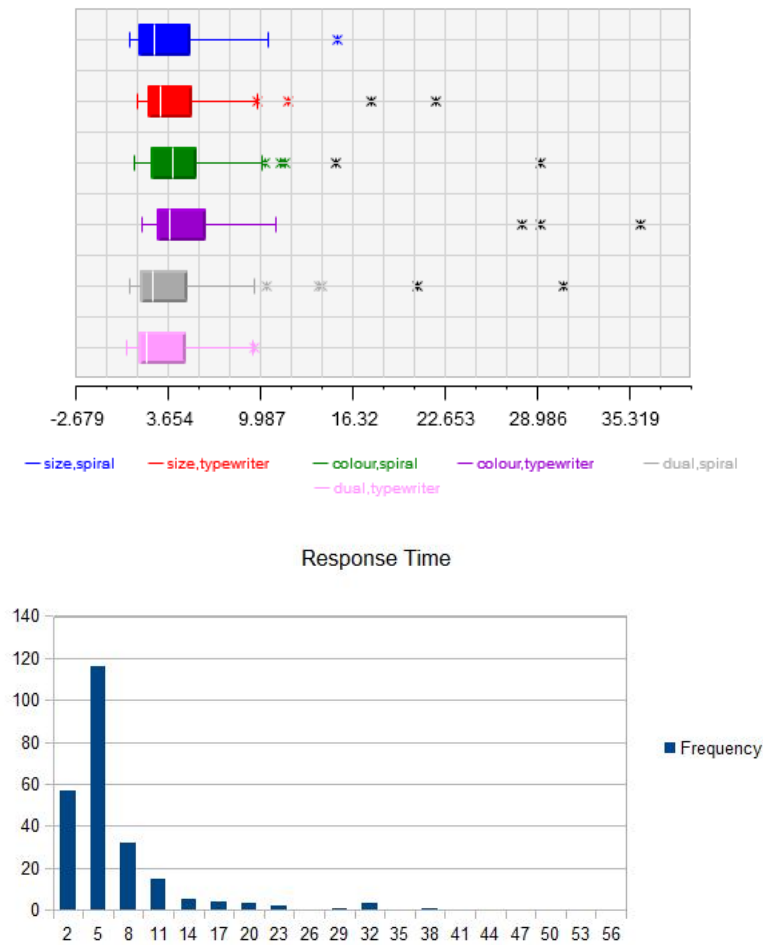


Figure 9.8: *Distribution of data*

Eye-gaze data for all outlying data points was also analysed. Analysis of eye-gaze data concluded participants used primarily the mapping visual feature to aid their search for the target tag (§9.10.1). However, when presented with a larger number of coloured targets such as in trial repetition one, participants appeared to resort to using a random search method or serial scanning to locate the target.

The underlying dataset used in repetition two (containing the most outliers) had a target category with only 21 tags. Analysis of the dataset revealed a longer than average character length (11.46 characters compared to a range of 6.55–7.69 characters for the other three datasets). Total characters in the dataset numbered 1146 compared to 722, 769 and 655 for the other datasets. The greater number of characters in the underlying dataset meant the tag cloud took up more space proportionally on the canvas than tag clouds from other repetitions. With the larger tag cloud filling up the canvas, participants appeared to be more likely to switch between search methods.

Figure 9.9 shows the eye-gaze data for two response time outlying values within repetition two. In this case participants applied a feature search with fixations clustered around tags with a large font size, but it appears that the larger tag cloud size resulted in making the search take longer than we might have expected (despite the target tag being in a potentially favourable location close to the average first fixation point).



Figure 9.9: Gaze data for repetition two: size mapping, with spiral layout, response times are outlying values

Analysis of the response time data revealed the presence of several outliers and strongly positively skewed data. Careful examination of eye-tracking data showed outliers were due to a variance in either category size, or average tag length which increased the complexity of the task for the participant, rather than experiment procedure or participant error.

9.10.3 Significance testing

A two-way ANOVA with repeated measures on the response times data showed a significant main effect for mapping ($F_{2,78} = 7.93$, $p < 0.000733$). Bonferroni-corrected pairwise comparisons showed the dual mapping of colour and size produced significantly faster visual search response times than singular size mapping ($t(39) = 2.66$, $p < 0.0113$) and singular colour mapping ($t(39) = 3.93$, $p < 0.0003$). This effect was achieved only when using the typewriter layout. Differences in search response time for mappings were not statistically significant in spiral layout, nor were the differences between singular size or colour mappings for either layout. All statistical testing was performed applying a logarithm transform so ANOVA data normality assumptions could be met. See Figure 9.10 for normalised response times for typewriter and spiral layout data.

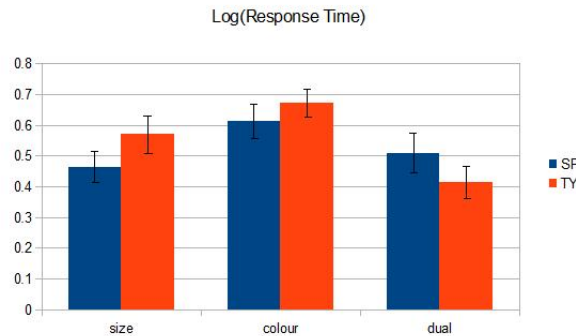


Figure 9.10: *Normalised visual search response time*

9.11 Threats to validity

As in experiment one, there is the possibility that usage of genuine categorical datasets introduced bias by increasing the tag size of some tags through number of characters (this has been shown to have an effect on user perception of tag importance [Bateman et al., 2008]). However, users were not identifying tags of subjective importance but searching for a specifically named tag. One dataset used (in trial repetition two) contained tags which had a larger average character length than datasets used in other repetitions. In experiment two, this resulted in

that repetition containing a number of outliers with significantly longer response times to target selection. This dataset was used across all conditions so this shouldn't affect the comparison of the outcomes between conditions.

Some of the experiment conditions have (averaged across across all stimuli shown) an overall potentially more favourable target location than other conditions, although it should be noted it is not clear exactly what functional relationship there is between the average first fixation point and location of the target. This does not appear to have resulted in a clear pattern in the response times or impacted the conditions where outliers have appeared — in fact, more outliers appeared in groups with favourable or mildly favourable target locations.

9.12 Summary and discussion

Analysis of the visual search response time data revealed a difference between singular and dual mappings for typewriter layout. As with experiment one, a number of outlying data points were noted across the conditions. In experiment one the outlying data values were confined mostly to two repetitions, and were related to a large target category size (for one dataset, and the other dataset became an accidentally large target category size due to inappropriate colour palette choices).

In experiment two there were also two repetitions which contained most of the outlying data values – one with a dataset with a large target category size, and one with a dataset with a long average tag length. It seems the long average tag length of this particular dataset was not a problem in experiment one when the target category size was very small (nine tags) but in experiment two, it suddenly became a problem when the category size was increased to 21. The extra number of characters in that particular dataset created a larger tag cloud filling up the canvas, and seemed to make it harder for people to find one individual tag (when they were searching for it within a larger target group of tags).

The dataset was used across all conditions (as was the large target category size dataset), but has caused some variability in the data, and probably contributed to the overall significant positive skew found in the data distribution. Therefore, like experiment one, all statistical testing was completed using a log-

arithm transform so data would conform to statistical testing requirements of normality.

H1: *Mapping a data field to both tag font size and colour leads to faster visual search times than mapping a data field to either tag font size or colour separately, when the tag cloud was displayed in a typewriter layout.* Differences in search response time for mappings were not statistically significant in spiral layout, nor were the differences between singular size or colour mappings for either layout. As in experiment one, results were different for both layouts, and the reasons for this remain unclear. Further close analysis of the eye-tracking data could be performed in the future to explore this further.

Supporting the analysis of eye-tracking data in experiment one, the eye-gaze data for experiment two also indicates the introduction of a visual property hint in tag searches can alter the search strategy from serial scanning or chaotic search methods, to the eye scan path focusing on tags with the target mapping.

9.13 Conclusion and future work

Our results indicate dual mappings of font size and colour as a data variable field can produce faster visual search response times than font size or colour alone. We think provision of multiple data mapping options to tag cloud visual properties in the Taggle interface highlights and reinforces the data mappings: this may help users identify data correlations and explore the data more effectively.

As with experiment one (Chapter 8), experiment two produced different results between the spiral and typewriter layouts, and future experimentation may help discover reasons behind this.

Varying target category size and average tag length over repetitions produced some abnormally long response times (outliers) for categories with greater sizes/tag lengths. Supporting our work from experiment one, eye-tracking data analysis showed when elements such as target category size or tag length increase the complexity of the task, participants tend to employ combination search methods switching between visual feature search, serial scanning and chaotic search. Future experiments may focus on the relationship between search methods and task complexity.

10

Experiment Three: Knowledge Discovery

A series of experiments was conducted to explore the potentials and limitations of the interactive tag cloud visualisation tool Taggle. This chapter details the third experiment, where we examined data exploration and knowledge discovery support for software engineering data in Taggle. Research questions and goals for the study are presented in §10.1. The methodology of the evaluation is detailed from §10.2 through §10.7. Results from the response times and the eye-gaze data analysis are discussed in §10.8. Finally, results are summarised and conclusions are presented in §10.10 and §10.11.

10.1 Knowledge discovery support in Taggle

We wanted to investigate the applicability of the tag cloud visualisation technique for gaining insight into multi-dimensional software engineering data. The tag cloud visualisation tool Taggle includes a number of interactive features to enable rich data exploration which are detailed in Chapter 4. In this experiment

we focus on the effectiveness of using an interactive tag cloud tool such as Taggle to compete data exploration tasks on an unfamiliar software engineering dataset. In particular, we ask the user to freely use the tool’s features to discover information about a dataset which contains simulated quality assurance metrics for a popular opensource project. These metrics represent indicators for “codesmells” (see §2.1.5 and §2.3.2), whose values should give the user insight into areas of the source code that are candidates for refactoring and overall source code quality. We had the following research questions:

RQ: Does Taggle support data exploration and knowledge discovery (discovery of relevant information) for software engineering data? How does it do this?

RQ: Is the tool easy to learn? Can users manage to successfully complete typical software engineering tasks after a minimum or realistic amount of training?

To achieve these objectives, we conducted a empirical user study which involved 18 participants. To assess the effectiveness with which users were able to discover knowledge using Taggle, user completion of tasks attempting to find patterns in multidimensional software engineering data was monitored. For collection of data, we used both a questionnaire and recordings of eye-gaze data or mouse movements. We hoped to observe people analysing and interpreting data correctly by making good use of Taggle features, such as applying appropriate data mappings to visual properties such as font size and order.

10.2 Participants

We recruited 18 University of Canterbury students to participate in the study (age 18–35, one female, with no reported uncorrected vision problems). All had completed, or were completing stage two or three computer science university papers. Participants received a \$20 gift voucher for participating in the study.

10.3 Apparatus

Due to performance issues on one of the machines, the study was run with some participants utilising a Tobii T60 eye-tracking machine attached to the 17-inch

Table 10.1: *Features mimicking typical qualities of a metric dataset*

Feature	Details
Variable correlation	Strong correlation (95%) between large class and conditional complexity Medium correlation (75%) between feature envy and lazy class
Data distribution	Large class and conditional complexity have a positively skewed distribution
Outliers	Three obvious outliers in the correlation between feature envy and lazy class

LCD screen. Eye-gaze data in this case was collected with Tobii Studio 3.2.1 software using a minimum fixation duration of 60 milliseconds with eye movements of each participant calibrated to five points. Other participants' mouse movements were recorded utilising Camtasia Studio 7.1.1 on a 15-inch LCD screen.

10.4 Tag corpus

A software engineering dataset was partially artificially generated to present typical characteristics of software engineering metric data (see §6.3 for the rationale behind using a mixture of real and artificial data). The dataset contained 120 classes with package and class names taken from open source software project JUnit. Five data variables were created for each class; the class name (a string identifier), and four numeric fields representing code smells associated with that class (large class, conditional complexity, feature envy and lazy class). The code smell fields contained a number between 0–100, where 0 represented no code smell, and 100 represented a very smelly class. This smell intensity scale is an arguably fairly realistic dataset, for instance this could be produced if a team rated code during a sprint, then assessed aggregate data as part of a sprint refactoring. To mimic typical qualities of a software engineering dataset of this type, numeric data fields were generated with the features described in Table 10.1.

10.5 Procedure

After completing a demographic questionnaire and signing a consent form, five minutes of training describing tag cloud visualisation was given pre-experiment.

Ten minutes was then spent on a software tutorial, exploring a dataset from a generic domain with Taggle. As one of our research questions was to explore the ease of learning capability of Taggle, the amount of time given on software training was kept to a minimum. We hoped a short exploration of a known dataset (explained in the pre-experiment tag cloud training) in the tool would mimic a realistic training session given between colleagues.

After training, participants were asked to freely explore a software engineering dataset using Taggle, and to use this exploration to attempt to complete a set of benchmark tasks. The tasks highlighted particular software engineering dataset characteristics such as correlations, outliers and distribution. These tasks are representative of a sample taskset utilised in analysing a software metric dataset. Benchmark tasks were distributed across task types traditionally associated with data mining — summarisation, classification, clustering and association.

Following completion of the taskset, participants filled in a NASA Task Load Index [Hart and Stavenland \[1988\]](#) questionnaire to assess their task workload with the Taggle system on five 7-point scales. To minimise time and bias, participants performed either experiment one or two and then three together. Overall, participation in both experiments took 40 to 60 minutes.

10.6 Tasks

Nine benchmark tasks were created to specifically explore the typical software engineering metric dataset features ([Table 10.1 on the preceding page](#)) that were contained in the target dataset. These were categorised across a subset of task types typically associated with data mining — summarisation, classification, clustering and association (trend analysis tasks usually were not included as the target dataset did not contain temporal data). [Table 10.2 on the next page](#) describes the benchmark tasks and task types.

Table 10.2: *Benchmark Tasks*

	Task Description	Task Type
T1	What class has the smelliest lazy class codesmell?	Classifying
T2	What codesmells does class <code>org.junit.runners.model.InitializationError</code> have?	Classifying
T3	What class has no smell for feature envy?	Classifying
T4	What can you tell me about the classes from package <code>org.junit.runners.model</code> ?	Clustering
T5	What can you tell me about the classes which contain low smelliness levels of lazy class?	Clustering
T6	For which codesmells can you establish a relationship? What is the strength of the relationship?	Associative
T7	What can you tell me about the distribution of values for codesmell large class ie. normal, skewed?	Summarising
T8	Are there classes which exhibit out-of-place values (outliers) for the correlation between codesmell feature envy and lazy class?	Summarising
T9	Which class — <code>junit.framework.JUnit4TestAdapter</code> or <code>org.junit.experimental.theories.Theory</code> — is smellier overall?	Classifying

10.7 Measurements

The dependent measures were accuracy in task completion (measured as the percentage of correct answers given), and experienced mental workload (measured by NASA’s task load index questionnaire).

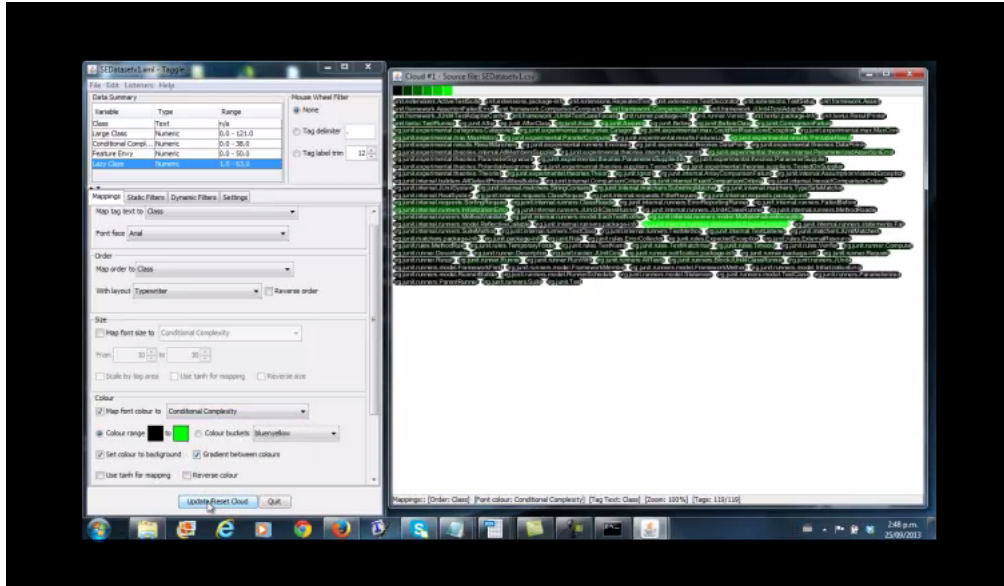
10.8 Results

10.8.1 Eye-gaze and mouse-tracking data analysis

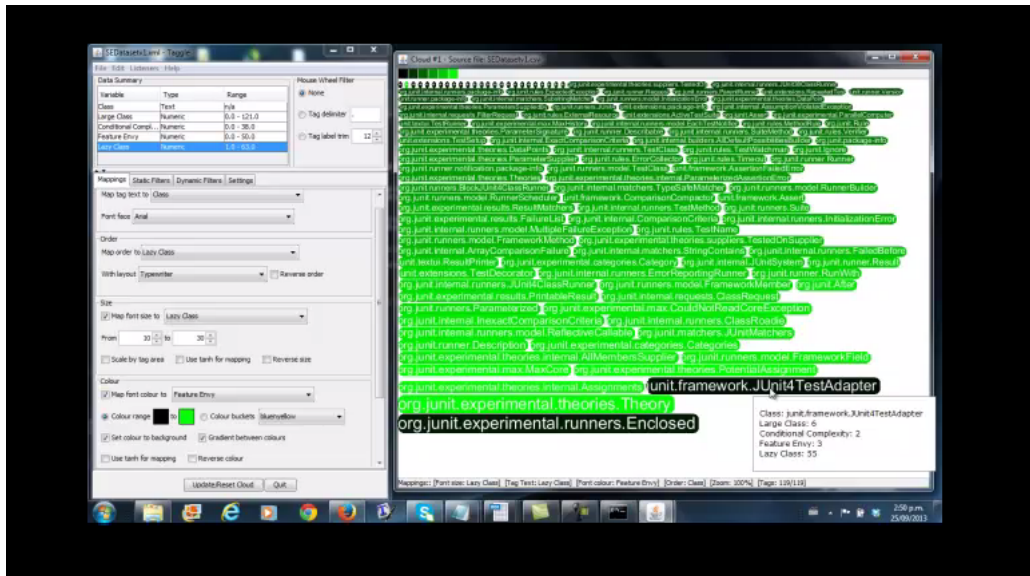
The analysis of the eye-tracking data was performed in two ways — in an exploratory manner looking for typical patterns of usage for various areas of the interface, and using Tobii Studio automatically calculated eye measurement metrics for dynamic areas of interest in the interface. Selected segments from recordings of individual participants in this experiment (including eye-gaze data or mouse movements) can be viewed at <http://www.cosc.canterbury.ac.nz/research/RG/svg/taggle/>.

Mapping selections Users tended to map data fields to tag order, font size and colour to answer questions — although some participants did try the transparency mapping also. Generally, participants used the default colours of black to green only and used the default setting of background colour with the tag mappings. Participants used mappings of font size against colour to complete tasks such as T6 relating to data correlations and discovery of outliers (T8). Successful discovery of data distributions for variables occurred by unmapping font size visual properties to get a clear picture of the distribution of colour across the tags. Examples of mappings to discover correlations, outliers and data distributions may be seen in Figure 10.1 on the following page.

Legends, helpers and summary information We were interested in finding out more about particular areas of the interface which provided information to help the user in interpreting the visual encoding — how much attention was paid to them, and how they were used. Those areas are as follows: *legend* (the colour chip legend in the top left hand corner of the visual encoding, showing a sample



(a) Participant finding the data distribution for conditional complexity field, question T7



(b) Participant finding outliers in correlation between lazy class and feature envy, questions T6 and T8

Figure 10.1: Participants mapping font size and tag background colour to complete tasks

colour for a variable numeric value), *status bar* (the status bar at the bottom of the canvas showing what fields are mapped to which visual properties), *details*

have been required to look at the legend to find out which colour was mapped to which category. The data summary screen received the most attention with an average of 20 fixations within 13 visits. It appears users initially spent a lot of time assessing the data fields available within the dataset.

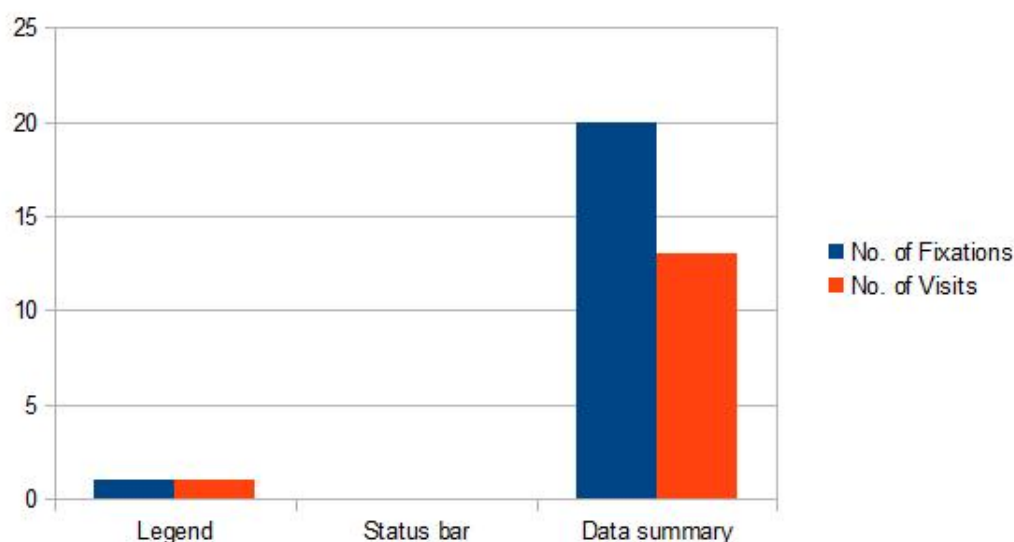


Figure 10.3: *Average number of fixations and number of visits to area of interest*

Figure 10.4 on the next page shows the participant average of total duration of visits in seconds to the area of interest. In agreement with the number of fixations and durations spent in each area, the legend received an average of 0.28 seconds for all visits, the data summary screen received 4.87 seconds and the status bar received no visits.

Details on demand pop-ups are different from the other three information sources in that some user action is required to use them whereas the others are nearly always visible (unless the data summary screen is contracted). Therefore, eye-gaze data for usage of pop-up details on demand had to be assessed manually. These pop-ups appear to be used much more frequently than any of the other three information sources. Typical pop-up frequency (calculated from an indicative sample) is around four per minute, or about 75 for the whole of completion of experiment three. We suspect that the use of some of these are “accidental” —

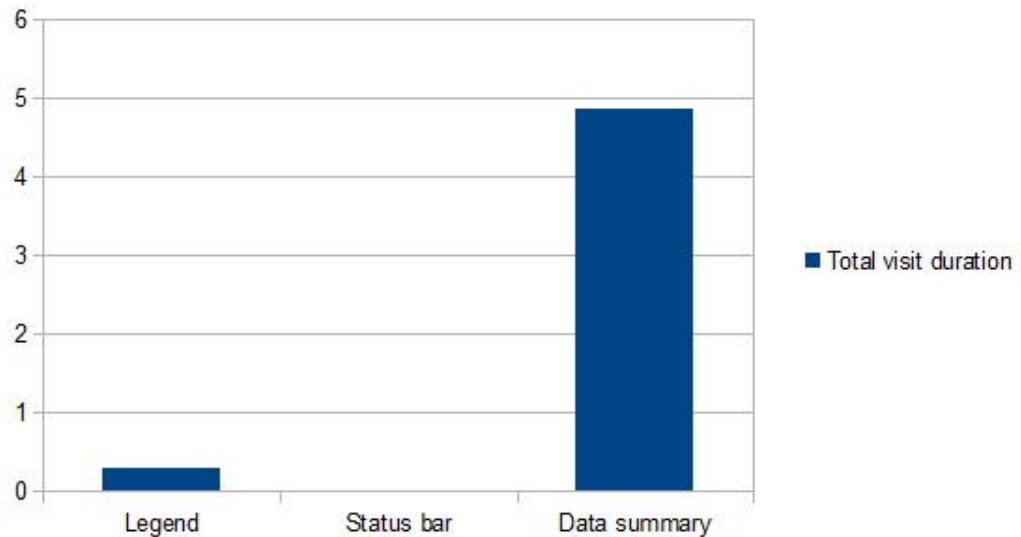


Figure 10.4: *Average total duration of visits to area of interest (in seconds)*

this is because the default mouse mappings cause the pop-up to appear whenever the mouse rests over a tag. (Initially we tried an explicit right-click action to display this pop-up, but during the heuristic evaluation detailed in Chapter 7 participants found it difficult to remember how to access this information). For future work, we might consider logging the pop-up duration in order to distinguish deliberate and accidental pop-up display. Details on demand pop-ups were used particularly during classifying tasks such as T9 where users were asked to compare overall smelliness of particular classes (see Figure 10.5 on the following page for examples) — pop-up usage helped users discover information about other smells associated with the classes they were interested in.

Filtering The mouse wheel filtering was used by most participants. This was used to apply a tag delimiter filter to strip out package names of the classes and make the tag cloud size more manageable. In Figure 10.6 on page 163, the participant is in the process of applying the mouse wheel filter.

Participants applied static filtering for questions which required specific information about tags, such as in clustering question T4 where users had to find

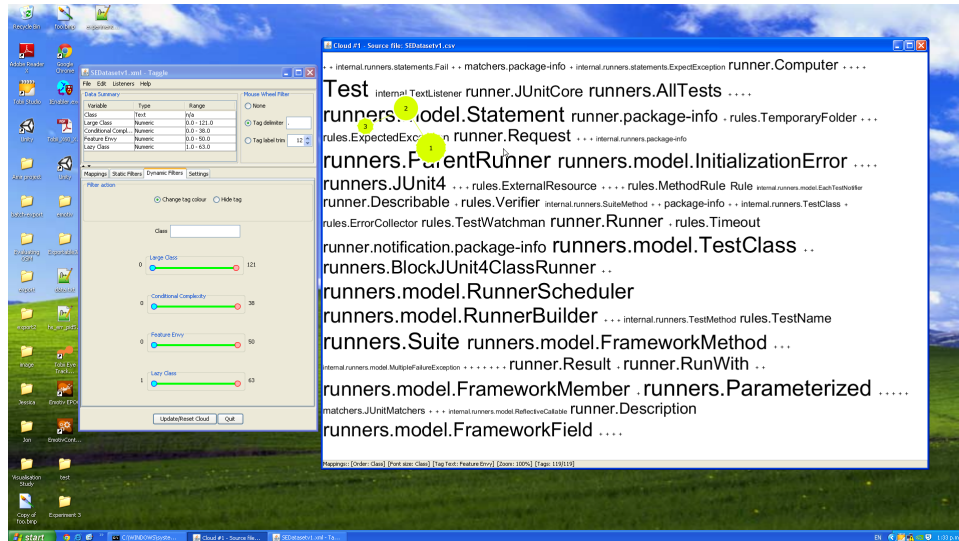


Figure 10.6: Participant is in the process of applying the mouse wheel filter to strip out package names

information about classes from a particular package. Figure 10.7 shows a participant applying a static filter for question T4.

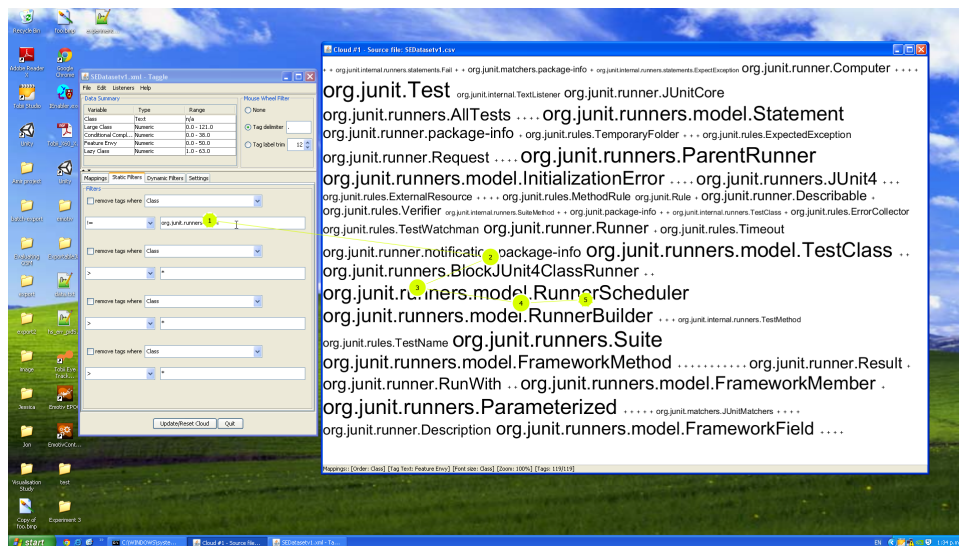


Figure 10.7: Participant has applied a static filter to locate particular classes in a package for question T4

Dynamic filtering was also applied by several participants when trying to find correlations between variables (T6), and when discovering information about

classes with low values of a particular smell (T5 — see Figure 10.8).

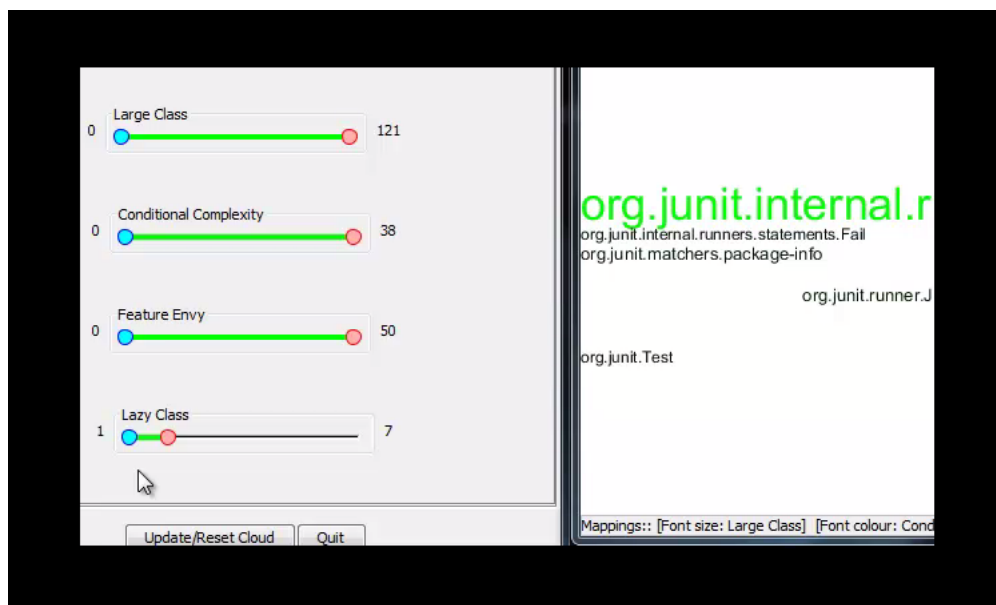


Figure 10.8: *Participant has applied a dynamic filter to locate classes low values of lazy class smell for question T5*

10.8.2 Task analysis

We defined three categories of answers for the tasks:

Correct answers: the participant has given all correctly named classes, or explanations/illustrations of relationships or patterns found.

Partially correct answers: the participant has given a subset of correctly named classes, or explanations/illustrations of relationships or patterns found.

Incorrect/unanswered: the participant has not answered the question, or has given an incorrect class name/explanation of relationships found.

We interpreted the percentage of correct and partially correct answers of the participants as the extent to which the system is effective in facilitating data exploration and knowledge discovery in multidimensional software engineering data.

The numbers of correct, partially correct and incorrect answers for each task is presented in Table 10.3 on the following page and Figure 10.9 on the next page shows the percentage of correct and partially correct answers for each task. Only T5 and T7 had percentages of correct/partially correct answers below 80 percent. T1, identifying the class with the smelliest lazy class codesmell, was answered correctly by 83 percent of participants. Nearly 95 percent of participants were able to correctly identify at least one codesmell that a particular class had, and identify what class had no smell for feature envy.

T4 was a more open-ended question, asking participants to state something about a particular package. In this case, the expected answer was to identify that classes from that package had very low values (0–1) for code smells large class and conditional complexity. Participants were marked with a partially correct answer if they identified the package had low values for at least one of large class or conditional complexity. Over 80 percent of participants were able to identify at least one of the two low smell levels.

T5 was also an open-ended question which did not give the participants an exact description of the sort of answer that we expected. Participants were asked to state something about classes which contain low smelliness levels of lazy class. The expected response was to note they also contained low levels of feature envy. This led nicely to question T6 which queried as to what relationships they could find between variables. T5 had the lowest overall correct completion rate of all the tasks with only 66 percent answering correctly, while nearly 90 percent were able to identify at least one correlation between variables. Interestingly, some participants were able to identify the correlation between feature envy and lazy class without noticing that classes with low lazy class values had low feature envy values. It is possible question T5 was worded in a way that confused some participants.

T7 also had a lower rate of correct answers with only 66 percent of participants able to correctly identify a skewed data distribution of values for code smell large class. Ninety-five percent of participants were able to identify outliers in a correlation between two variables — even participants who hadn't been able to identify any variable correlations in question T6. Finally, nearly 90 percent of participants were able to correctly compare two classes and identify which class

Table 10.3: *Number of correct, partially correct and incorrect answers*

Task	Task Type	Correct	Partially correct	Incorrect
T1	Classifying	15	0	3
T2	Classifying	16	1	1
T3	Classifying	17	0	1
T4	Clustering	10	5	3
T5	Clustering	9	3	6
T6	Associative	16	0	2
T7	Summarising	12	0	6
T8	Summarising	17	0	1
T9	Classifying	16	0	2

was smelliest overall.

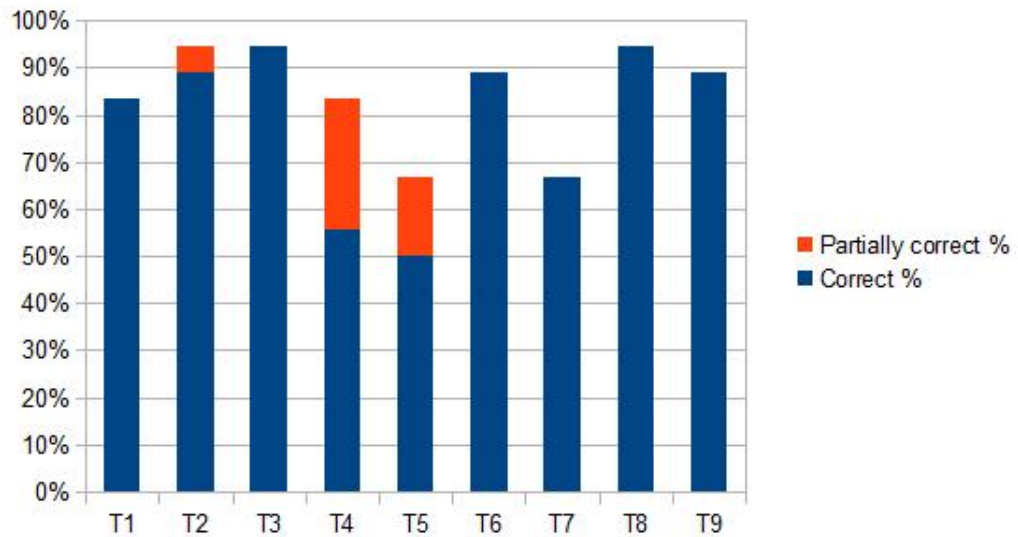


Figure 10.9: *Percentage of correct and partially correct answers per task*

Figure 10.10 on the following page shows the percentage of correct and partially correct answers for each task type category. The clustering tasks had the lowest overall percentage of correct answers as well the highest number of answers which were only partially correct.

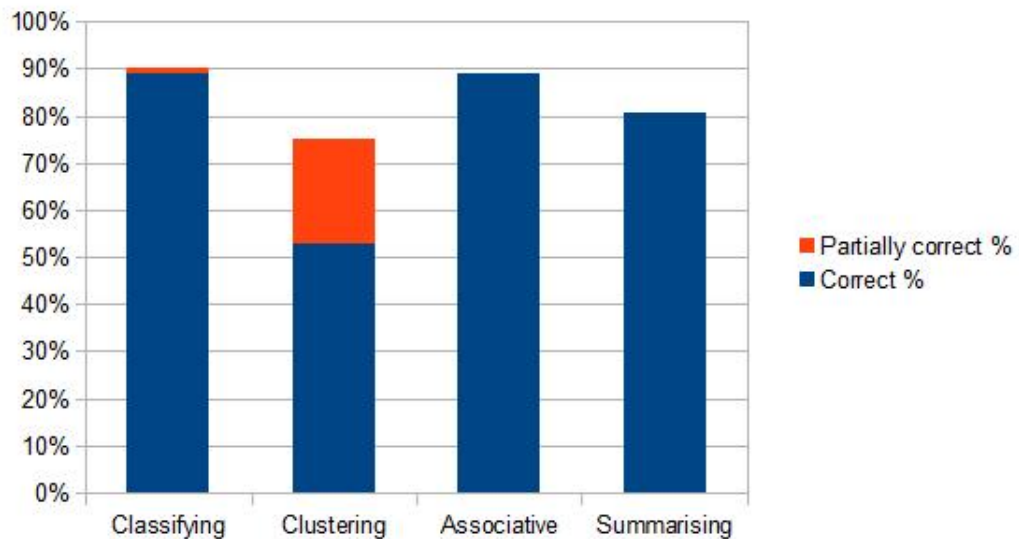


Figure 10.10: *Percentage of correct and partially correct answers per task type*

10.9 Workload measures

The NASA-TLX[Hart and Stavenland, 1988] questionnaire is a multidimensional assessment used to rate perceived workload on six scales: mental demand, physical demand, temporal demand, performance, effort, and frustration. The scales are presented to participants as five 7-point scales with increments of high, medium and low estimates for each point (resulting in 21 gradations on the scales). This questionnaire was used to acquire subjective feedback on the perceived mental workload of the software engineering tasks. Figure 10.11 on the next page shows the averaged results presented on a 7-point scale — physical demand was perceived as very low (0.84), while mental demand and effort were the highest, rated 3.46 and 3.60 respectively. It can be noted that the averaged task indexes were perceived in the lower half of the scales for all workloads.

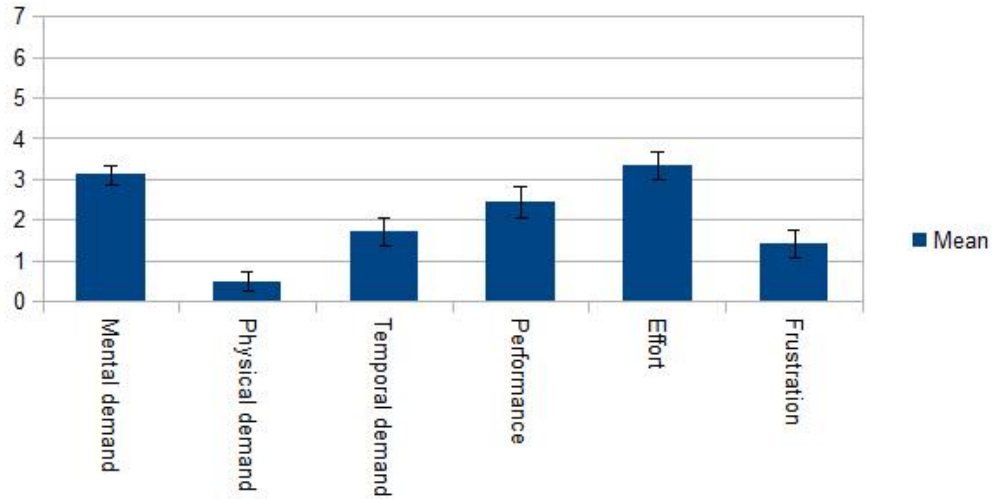


Figure 10.11: *NASA Task Load Index: perceived workload of Taggle on a 7-point scale (lower is better)*

10.10 Summary and discussion

Analysis of participant percentages of correct and partially correct answers for each task found that all tasks were completed between a 66 percent and 94 percent completion rate. Tasks which were performed the most poorly were tasks which asked the user to find classes which contained low levels of a particular code smell (T5) and discovery of the data distribution of another code smell (T7). Although this was briefly discussed during the software training, we query whether all participants had an understanding of the terms used in the questionnaire — ‘skewed’ and ‘normal’ data distributions. The task which had the highest percentage of partially correct answers was T4, where participants were asked an open-ended question to state something about classes from a particular passage.

An analysis of tasks categorised into typical data mining categories (classification, clustering, association, summarising) found that clustering tasks performed the worst at a 74 percent successful completion rate, with the highest percentage of partially correct answers. We attribute this to the two clustering task questions being more open-ended than other questions (for example, ‘what can you tell me about classes from package ...’). This was because our goal was to see if

participants could identify some similar characteristic of a group of tags. This open-ended nature of these questions resulted in some variation in answers. Both classifying and associative tasks had the highest percentage of successful answers at nearly 90 percent.

Eye-gaze and mouse-tracking data were analysed to find out how participants utilised the features in Taggle to complete tasks. Exploratory analysis showed participants who were able to successfully complete tasks used the rich interactive features in Taggle such as mapping multiple visual properties to data fields, and static and dynamic filtering. Participants who were not successfully able to complete a task tended to use inappropriate data mappings. Parts of the interface which showed summary data and information to help the user interpret the visual encoding were analysed to find out whether participants found them useful in the data exploration process. The data summary panel was identified as being highly used early in the exploration process to gain an overview of the underlying dataset. The data summary panel was included by request following results from our previous heuristic evaluation (see Chapter 7). The mapping status bar was unused and the colour chip legend was used minimally. We expect that had the target dataset included categorical data, the colour chip legend would have been used more extensively so that users could find out which colours were mapped to particular categories. Pop-up details on demand for individual tags were also widely used for comparative or classifying tasks such as T9.

RQ: *Does Taggle support data exploration and knowledge discovery (discovery of relevant information) for software engineering data? How does it do this?*

Overall, the task analysis and eye-tracking results were encouraging. Users had a generally high completion rate for all tasks (66 percent and over) and eye-gaze data showed users analysing and interpreting data using Taggle mappings for visual properties such as font size and colour. Interactive features such as filtering mechanisms to enable rich data exploration were also used.

RQ: *Is the tool easy to learn? Can users manage to successfully complete typical software engineering tasks after a minimum or realistic amount of training?*

We think the generally high task completion rates show that it is possible to

take an interactive tag cloud visualisation tool such as Taggle, and have participants discover information about an unfamiliar dataset with a minimal ten minute training session.

The study participants also completed a subjective assessment used to rate perceived workload (NASA-TLX). Averaged task indexes were perceived in the lower half of the 7-point scales for all workloads.

10.11 Conclusion and future work

Our results indicate the Taggle interactive tag cloud visualisation tool can be used by people with minimal training to discover relevant information about an unknown software engineering dataset. Experiment participants were able to effectively complete visual classification, clustering, association and summarising tasks to a generally high standard. Eye-gaze and mouse-tracking data showed participants utilising the interactive features of Taggle to analyse and interpret data. The results of this study throw some interesting questions which could initiate future work — further exploring the idea that a tag cloud interactive tool is easy to learn, including temporal data in our investigations, and looking into the usefulness of the data summary panel and details on demand pop-up in helping the user interpret the visual encoding.

11

Conclusions

There is a need in software engineering to explore new visualisation techniques with a low-level of conceptual complexity, and for the effectiveness of visualisation tools to be adequately quantified. This thesis investigated the application of the tag cloud metaphor to the software engineering domain and identified limitations in current research evaluating tag cloud visualisation; a lack of research evaluating visualisation tools and techniques for domains outside of the web and user-generated data domains, and a limited range of evaluation approaches. In particular, there is a dearth of research utilising evaluation strategies which focus on visual data analysis and reasoning, or collaborative data analysis. These strategies have a goal of understanding the underlying data exploration process of a tool (as opposed to usability of a system slice or technique), and are of high relevance and practical value in the information visualisation field. This is also of special importance in software engineering, as demonstrations of visualisation benefits exploring data using realistic scenarios may be a key factor contributing to the lack of visualisation techniques being integrated into mainstream development environments. In response, this thesis contributes a new system for interacting with software engineering or other multi-variate data using tag clouds, and evaluates both utilised enhanced tag cloud features, and the knowledge discovery

process within the system itself.

11.1 Review of thesis contributions

The central theme in this thesis is the design and evaluation of a visualisation system that utilises tag clouds, a highly recognisable visualisation with a low-level of conceptual complexity, and that supports exploration of multi-variate data such as that found in software engineering. To that end, we analysed the challenges in visualising multi-variate data, and the capabilities of tag clouds (Chapters 3) — in particular necessary task types and the effects on user perception for available visual properties — to develop design considerations guiding the development of an interactive tag cloud visualisation system.

We applied these considerations to the design of Taggle (Chapter 4), a Java-based tag cloud visualisation system utilising enhanced tag cloud features to explore multi-variate software measurements. We conducted a heuristic evaluation of Taggle using domain experts (Chapter 7). This revealed that the tag cloud technique of contrasting visual font properties mapped to data fields was felt to be instinctively comprehensible, and underlying data distributions, outliers and tags clusters were able to be inferred. However, it was considered the amount of information that could be interpreted was greatly dependent on the software’s support for selection of appropriate mappings. Following the heuristic evaluation and before the experiments, alterations were made to the software to improve the system’s usability and appropriateness of exploration of multi-variate data. Thus the experiment subjects gained the benefit of the suggested improvements.

We performed a systematic mapping study synthesising existing tag cloud evaluation research (Chapter 5). This served to provide an overview of existing evaluations of the tag cloud visualisation and tools which incorporated the technique, and mapped evaluation approaches and methods, as well as the target domain. We discovered most research focused on interactive interfaces for special datasets, mediums or populations but utilised a limited range of evaluation approaches – focusing on *visualisation use* strategies which measured timed user performance or subjective experience, as opposed to insight or knowledge gath-

ering support. No interface identified in the mapping study proposed a system such as Taggle, where data fields from a multi-variate dataset are mapped to tag cloud visual properties and manipulated interactively. Tag cloud visualisation itself had not been as extensively evaluated as other areas, indicating there was still room to define their overall effectiveness and develop ways to improve the tag cloud as a technique. Furthermore, research was significantly skewed towards web and user generated data domains, with only one paper evaluating a tag cloud visualisation system in the software engineering field.

Based on the systematic mapping study results, a series of targeted evaluations was planned and conducted to explore the potentials and limitations of Taggle (Chapter 6). In order to obtain a broad-based investigation of points of relevance for both the tag cloud technique and our interactive tool, the experiments were conducted in both areas of *visualisation use* (the tag cloud technique) and *data process analysis* (a whole-tool approach focused on the knowledge discovery process). Two *user performance* experiments were conducted where we examined the enhanced tag cloud features included in Taggle to see if they improved user performance in visual search tasks (Chapters 8 and 9). One *visual data analysis and reasoning* evaluation was conducted where we examined data exploration and knowledge discovery support in Taggle (Chapter 10).

Tag background colour We compared user visual search response times alternating foreground and background colour in target tags. Results indicated usage of tag background colour as a data variable field can produce faster search response times than foreground colour when the target tag is small. Small tags are inevitable in tag cloud visualisation when using a full range of tag sizes, and when visualising large datasets, such as those found in software engineering. We think providing the background colour option as the default setting in Taggle may help users identify mapped variables (particularly for small or iconified tags), and explore correlations more efficiently between data variables represented by size and colour.

Dual data mappings We compared user visual search response times alternating single mappings (colour or font size) and dual mappings (colour and font size

together) in target tags. Results indicated dual mappings of font size and colour to a data variable field can produce faster visual search response times than font size or colour alone. We think provision of multiple data mapping options to tag cloud visual properties in the Taggle interface highlights and reinforces the data mappings: this may help users identify data correlations and explore the data more effectively.

Visual search patterns within a tag cloud Eye-gaze data was collected from experiments where participants were executing a visual search within a tag cloud. Our eye-tracking data analysis showed that the introduction of a visual property hint when performing a search task can alter the search strategy to the eye scan path focusing on tags with the target mapping (efficient feature search). When elements such as target category size or tag length increased the complexity of the task, users generally employed combination visual search methods in a tag cloud, switching between visual feature search, serial scanning and chaotic search methods.

Knowledge discovery We evaluated Taggle’s support for data exploration and knowledge discovery through an empirical user study, where participants explored an unknown software engineering dataset and attempted to complete a set of benchmark tasks. Results indicated Taggle could be successfully used by people with minimal training to discover relevant information in a dataset. Experiment participants were able to effectively complete visual classification, clustering, association and summarising tasks. Eye-gaze and mouse-tracking data showed participants utilising the data summary panel and features such as mapping multiple visual properties to data fields, and static and dynamic filtering to analyse and interpret data.

11.2 Limitations and future work

We hope this thesis will serve as a prelude to a continuing stream of research investigating the limits and potentials of tag clouds in both general information

and software engineering visualisation. In this section we elaborate some of the limitations of our experiments and opportunities for future research.

Spiral versus typewriter layout Both *visualisation use* experiments on a static tag cloud produced irregular results between the spiral and typewriter layouts and we were unable to determine the reasons behind this. It is possible that there are inherent differences in the way users search for visual targets in a spiral layout, or it may be something particular to the spiral layout algorithm used in Taggle. Future eye-tracking data analysis or experiments may focus on discovering more about this difference.

Visual search patterns and task complexity When elements such as target category size or tag length increased the complexity of the task, users generally employed combination visual search methods in a tag cloud. For example, varying target category size over experiment repetitions produced some abnormally long visual search response times for categories with greater sizes. It would be interesting to focus on category size (or task complexity) as an experiment factor in exploring eye-scanning methods and to discover at what point (such as overall percentage or specific number of tags) participants start producing atypical response times. Response times for varying category or dataset sizes are particularly pertinent in the software engineering domain where dataset size can be a challenging element.

Temporal data A limitation of our knowledge discovery experimentation with Taggle is that temporal data was absent and therefore trend analysis was not included with the other benchmark task types. There are a variety of ways in which temporal data can be analysed using tag cloud visualisation (for example comparing multiple clouds or using visual properties such as order to display time, as shown in [Figure 3.7 on page 46](#), or evolving clouds that dynamically change in real time). The effectiveness of tag cloud visualisation for trend analysis is yet to be determined.

Use of interface features to assist mapping choices Exploratory analysis of eye-gaze data for the knowledge discovery experiment showed the summary data screen was used extensively. Other information displaying parts of the

interface such as the colour chip legend and status bar were used minimally or not at all. This could drive another round of interface refinement such as the one that happened after the heuristic evaluation. Further analysis of the eye-gaze data may be done for other interface features as well as to look carefully at the process of how the summary data screen was used for tasks.

Taggle comparative evaluation We can use the same data, taskset and NASA-TLX questionnaire to gather results from another information visualisation tool. These results can be used in order to compare the task completion and perceived workload of Taggle to other software engineering visualisation tools.

11.3 Closing remarks

Visualisation allows us to gain insight into the screeds of information which are available to us in ever-increasing amounts. Modern software has an inherent scale and complexity, constantly evolving, with complex relationships between components. Despite the plethora of techniques proposed for visualising software, they have not been widely applied or integrated into mainstream development environments. In software engineering, there is a need to explore new visualisation techniques — particularly those such as tag clouds which have an apparent low-level of conceptual complexity, and furthermore, for the effectiveness of such techniques and tools to be demonstrated. This thesis has presented *Taggle*, an interactive visualisation tool utilising tag clouds. Through careful consideration of applicable perceptual factors, Taggle is particularly suited to software engineering data. Empirical evidence suggests the system is useful for visual classification, clustering, association and summarising tasks. This thesis illustrates the greater potentials of the common tag cloud in exploring and making sense of multi-variate data.

Appendices

A Software engineering data

A.1 Eclipse metrics plugins

- *Metrics*¹: Discontinued. Contains CK metric suite and others.
- *Metrics2*²: Continuation of above metrics project.
- *Google CodePro AnalytiX*³: Contains a variety of metrics but does not appear to contain CK metric suite. Contains Halstead Science metrics.
- *GERT*(*Good Enough Reliability Tool*) (2006): Developed at the North Carolina State University, this plugin is using the STREW metric suite, containing a number of complexity, OO and size adjustment metrics.
- *Checkstyle*

A.2 Software analysis platforms

- *Alitheia Core*⁴: Developed at Athens University, this is a platform for software quality analysis designed for research on large data sources. Works by importing source, mailing lists, bugs etc into a database and doing data preprocessing and metadata extraction. Researchers create analysis plugins by implementing an interface.
- *MASU platform*⁵: Developed at Osaka University, this analysis platform calculates CK Metric suite and cyclomatic complexity metrics for a variety of different programming languages.
- *Sonar platform*^{6,7}: Sonar is an open-source web based code quality analysis tool for Maven-based Java projects. It covers a wide area of code quality

¹<http://metrics.sourceforge.net/>

²<http://metrics2.sourceforge.net/>

³<http://code.google.com/javadevtools/codepro/doc/index.html>

⁴<http://istlab.dmst.aueb.gr/~george/pubs/2009-ICSERD-GS/poster.pdf>

⁵<http://masu.sourceforge.net/>

⁶<http://www.sonarsource.org/>

⁷<http://docs.codehaus.org/display/SONAR/Documentation>

check points and is possible to extend via a plugin mechanism. Sonar already contains coverage clouds using class names in a limited capacity. It is possible to collect metrics generated by a 3rd party source and inject into Sonar to visualise. It is also possible to compile a custom metric.

- *Moose*¹
- *Borland Together*

A.3 Metric frameworks/tools

Command line tools or libraries that can generate metrics.

- *CKJM*² (2005): An open source java tool for calculating CK metrics suite, CA and NPM. This is a simple command line tool, and is also an ant task and maven plugin.
- *CKJM Pro*³ (2010): An extended version of CKJM which generates additional metrics.

Table 1: *Output format from tools*

Tool	Output
Sonar	PDF, HTML or CSV
CYVIS	CSV, XML
Rational software analyser	XML, PDF, HTML
Moose	CSV, XML
Eclipse metrics2	XML
RSM	HTML, CSV, XML
Borland Together	XML, HTML, CSV, Tab separated
SourceMonitor	XML, CSV
CKJM	Space separated
State of Flow	HTML, CSV, XML

¹<http://www.moosetechnology.org/>

²<http://www.spinellis.gr/sw/ckjm/>

³http://gromit.iiar.pwr.wroc.pl/p_inf/ckjm/

Table 2: *Static and dynamic analysis tools*

Tool type	Tool
Code coverage tools	Cobertura, NCover
Memory analysis	Eclipse Memory Analyzer
Performance analysis	DTrace
Static analysis	Checkstyle, PMD and FindBugs

B Heuristic artefacts

B.1 Heuristic descriptions

Heuristic	Code
<p>INFORMATION CODING (VISUAL REPRESENTATION): The perception of information is directly dependent on the mapping of data elements to visual objects. This should be enhanced by using realistic characteristics/techniques or the use of additional symbols.</p> <p><i>Example guidelines:</i></p> <ul style="list-style-type: none">• Are the data to visual element mappings understandable and effective?• Is the use of additional symbols in the representation understandable and effective?• Can you get a general understanding of the underlying data values of an individual item and where the individual data item fits into the overall dataset?	B5

Continued on next page

Table 3 – *Continued from previous page*

Heuristic	Code
<p>MINIMAL ACTIONS: Workload with respect to the number of actions necessary to accomplish a task. It is a matter of limiting as much as possible the steps users must go through.</p> <p><i>Example guidelines:</i></p> <ul style="list-style-type: none"> • Are the number of steps required to perform a task reasonable? • For data entry, are currently defined default values displayed in their appropriate data fields? • Can the user directly go to a requested view, without having to go through intermediaries? 	E7
<p>FLEXIBILITY: Number of possible ways of achieving a given goal, means available for customisation to take into account working strategies, habits and task requirements.</p> <p><i>Example guidelines:</i></p> <ul style="list-style-type: none"> • Are users provided with sufficient means to control visualisation configuration? • Are users permitted to define, change or remove default values for settings? • When some displays are unnecessary, can users remove/hide them temporarily? • Can users change settings in any order? 	E11

Continued on next page

Table 3 – *Continued from previous page*

Heuristic	Code
<p>CONSISTENCY: Design choices are maintained in similar contexts and different when applied to different contexts.</p> <p><i>Example guidelines:</i></p> <ul style="list-style-type: none"> • Are window titles always located in the same place? • Are the configuration controls consistent? • Are similar procedures used to perform tasks? • Are labels (phrasing, punctuation, placement) consistent? 	E16
<p>RECOGNITION RATHER THAN RECALL: The user should not have to memorise a lot of information to carry out tasks.</p> <p><i>Example guidelines:</i></p> <ul style="list-style-type: none"> • Are the available user actions visible to the user? • Can the user perform tasks without having to recall information? • Can tasks be performed without referring to external documentation? 	C6

Continued on next page

Table 3 – *Continued from previous page*

Heuristic	Code
<p>SPATIAL ORGANISATION (VISUAL REPRESENTATION):</p> <p>User's orientation in the information space, distribution of elements in the layout, precision and legibility, efficiency in space usage and distortion of visual elements.</p> <p><i>Example guidelines:</i></p> <ul style="list-style-type: none"> • Can I easily locate an information element in the display? • Are the individual elements legible? • Is space used efficiently in the layout? • Am I aware of the overall distribution of information elements in the representation? • Are some objects occluded by others? • Does the layout follow a logical organisation? • Do I understand how tag placement is related to the selected layout and ordering? 	B3

Continued on next page

Table 3 – *Continued from previous page*

Heuristic	Code
<p>REMOVE THE EXTRANEIOUS: minimising the distracting effect of extra information when locating information, gaining an overview, or making a comparison.</p> <p><i>Example guidelines:</i></p> <ul style="list-style-type: none"> • Are means provided to reduce the distracting effect of extra information when locating information? • Are means provided to reduce the distracting effect of extra information when gaining an overview? • Are means provided to reduce the distracting effect of extra information when making a comparison? 	D10
<p>DATASET REDUCTION (INTERACTIVITY MECHANISM): Concerns the provided features for reducing a dataset, their efficiency and ease of use.</p> <p><i>Example guidelines:</i></p> <ul style="list-style-type: none"> • Are means provided to filter/reduce a dataset? • Are means provided to prune or cut off information that may be irrelevant? • Are filtering mechanisms efficient and easy to use? 	B9

Continued on next page

Table 3 – *Continued from previous page*

Heuristic	Code
<p>ORIENTATION AND HELP (INTERACTIVITY MECHANISM): Functions like support to control levels of details, redo/undo of actions and representing additional information.</p> <p><i>Example guidelines:</i></p> <ul style="list-style-type: none"> • Are means provided to control levels of detail? • Are means provided to redo/undo user actions? • Is requested additional information represented, or accessible? 	B7
<p>PROMPTING: Refers to the means available in order to guide the users towards making specific actions and means that help users to know the alternatives when several actions are possible depending on the contexts. Prompting also concerns status information, that is information about the actual state or context of the system, as well as information concerning help facilities and their accessibility.</p> <p><i>Example guidelines:</i></p> <ul style="list-style-type: none"> • Are users aware of valid values for interface options? • Are measurement units displayed for data entry? • Is status information displayed? • Are labels provided for all fields? • Are cues provided on the acceptable length of data entries? • Are titles provided for each window? • Is on-line help and guidance provided? 	E1

B.2 Heuristic checklist

Heuristic Checklist	Check
<div> <div>✓ - fulfils criteria</div> <div>✗ - doesn't fulfil criteria</div> <div>~ - somewhat fulfils criteria</div> </div>	
INFORMATION CODING – VISUAL REPRESENTATION: <ul style="list-style-type: none"> Are the data to visual element mappings understandable and effective? Is the use of additional symbols in the representation understandable and effective? Can you get a general understanding of the underlying data values of an individual item and where the individual data item fits into the overall data set? 	
MINIMAL ACTIONS - INTERACTIVITY: <ul style="list-style-type: none"> Are the number of steps required to perform a task reasonable? For data entry, are currently defined default values displayed in their appropriate data fields? Can the user directly go to a requested view, without having to go through intermediaries? 	
FLEXIBILITY - INTERACTIVITY: <ul style="list-style-type: none"> Are users provided with sufficient means to control visualisation configuration? Are users permitted to define, change or remove default values for settings? When some displays are unnecessary, can users remove/hide them temporarily? Can users change settings in any order? 	
CONSISTENCY – INTERACTIVITY: <ul style="list-style-type: none"> Are window titles always located in the same place? Are the configuration controls consistent? Are similar procedures used to perform tasks? Are labels (phrasing, punctuation, placement) consistent? 	
RECOGNITION RATHER THAN RECALL – INTERACTIVITY: <ul style="list-style-type: none"> Are the available user actions visible to the user? Can the user perform tasks without having to recall information? Can tasks be performed without referring to external documentation? 	

SPATIAL ORGANISATION – VISUAL REPRESENTATION: <ul style="list-style-type: none"> • Can I easily locate an information element in the display? • Are the individual elements legible? • Is space used efficiently in the layout? • Am I aware of the overall distribution of information elements in the representation? • Are some objects occluded by others? • Does the layout follow a logical organisation? • Do I understand how tag placement is related to the selected layout and ordering? 	
REMOVE THE EXTRANEOUS – INTERACTIVITY: <ul style="list-style-type: none"> • Are means provided to reduce the distracting effect of extra information when locating information? • Are means provided to reduce the distracting effect of extra information when gaining an overview? • Are means provided to reduce the distracting effect of extra information when making a comparison? 	
DATA SET REDUCTION – INTERACTIVITY: <ul style="list-style-type: none"> • Are means provided to filter/reduce a dataset? • Are means provided to “prune” or cut off information that may be irrelevant? • Are filtering mechanisms efficient and easy to use? 	
ORIENTATION AND HELP – INTERACTIVITY: <ul style="list-style-type: none"> • Are means provided to control levels of detail? • Are means provided to redo/undo user actions? • Is requested additional information represented, or accessible? 	
PROMPTING – INTERACTIVITY: <ul style="list-style-type: none"> • Are users aware of valid values for interface options? • Are measurement units displayed for data entry? • Is status information displayed? • Are labels provided for all fields? • Are cues provided on the acceptable length of data entries? • Are titles provided for each window? • Is on-line help and guidance provided? 	

Additional comments:

B.3 Typical usage patterns

Sample tasks for exploration of a dataset.

Set-up of mappings Note that mappings can be either NUMERICAL or ALPHABETICAL.

-
1. Map tag text to a variable e.g. ‘Baby name popularity’ dataset map tag text to variable ‘Name’.
 2. Map order (layout) to a variable e.g. ‘US popularity’. The data is laid out from smallest value to largest OR from largest value to smallest if ‘reverse order’ is checked. Choose a layout algorithm of typewriter or spiral. Use step layout to see how/where individual tag elements are placed.
 3. Map font size to a variable e.g. ‘US popularity’. The smallest font size will be mapped to smallest data values and largest font size to the largest data values.
 4. Map colour to a variable e.g. ‘NZ popularity’. The ‘from’ colour will be mapped to smallest data values and ‘to’ colour to the largest data values.
 5. Press ‘Update/reset cloud’ to produce a cloud.

Interacting with the data and filtering information

- Filter tags on a delimiter (mouse wheel scroll) e.g. filter out package names in a class
- Filter tag labels that are too long (mouse wheel scroll) e.g. filter agile story labels to only 6 characters
- Remove tags temporarily from dataset e.g. remove runners from a particular club
- Filter tags from a dataset, keeping them in view (dynamic filters) e.g. tag the tag colour of runners from a particular club
- When there are too many tags to display in the window, some of the smaller ones will be displayed with a + symbol
- Make a new cloud from a selected subset of tags
- Set the tag colour to be displayed in the background
- Compare tags by selecting and dragging next to one another

Exploring data and generating hypotheses

- Identify correlations in the data e.g. baby names that are popular in the NZ appear correlated with popular baby names in the US
- Outlier identification e.g. the estimate for this agile story is much higher than the other stories
- Identify clusters of data with similar characteristics e.g. runners from this club have a lower score
- Locate a particular data element e.g. find the most popular baby name in the US
- Compare multiple data elements e.g. which baby name is more popular, 'Shane' or 'Antonio'
- Identify the distribution of the data e.g. most users have a low average number of hours to fix a software bug

B.4 Usability issues found linked to heuristics

Table 4: *Mapping Selection/Cloud Creation*

Heuristic	Issue	Severity
E7 (minimal actions) E1 (prompting)	Need overview of data immediately when first open a dataset	3
E1 (prompting)	Mapping font size to text (alphabetical) isn't helpful	3
E1 (prompting)	Figuring out what mappings to use to initially very difficult	5
E1 (prompting)	There are too many variables to play with. Need to reduce at the start and pair up e.g. size/order. I cannot be sure if there is a relationship in the data or I just haven't found it yet. (Rephrased — I need guidance towards taking an appropriate action specific to my task).	4
E16 (consistency) D10 (remove the extraneous) E7 (minimal actions)	Require the mappings to be reversed if data hasn't been filtered correctly, like what you can do with order (Rephrased — I shouldn't have to have to edit my source file to reverse data).	3
E7 (minimal actions)	Having to click update/reset after each mapping change	2

Table 5: *Static and Dynamic Filtering*

Heuristic	Issue	Severity
E7 (minimal actions) B9 (dataset reduction)	It is is easy to forget the filter check box	3
B9 (dataset reduction)	It is not obvious how the dynamic filter worked and the red, green, blue colours were confusing	2
B9 (dataset reduction) E16 (prompting) E7 (minimal actions)	Don't reset mouse wheel when click update/reset cloud. Also I didn't know how to use it until explained	2

Table 6: *Information Coding and Spatial Organisation*

Heuristic	Issue	Severity
B5 (information coding)	One colour per class when mapping data to colours (colour legend)	2
B3 (spatial organisation)	Spiral is easier to see patches of clubs with high scores, don't get the disjoint at the end of each row in typewriter layout	3

C Eye-gaze visualisations

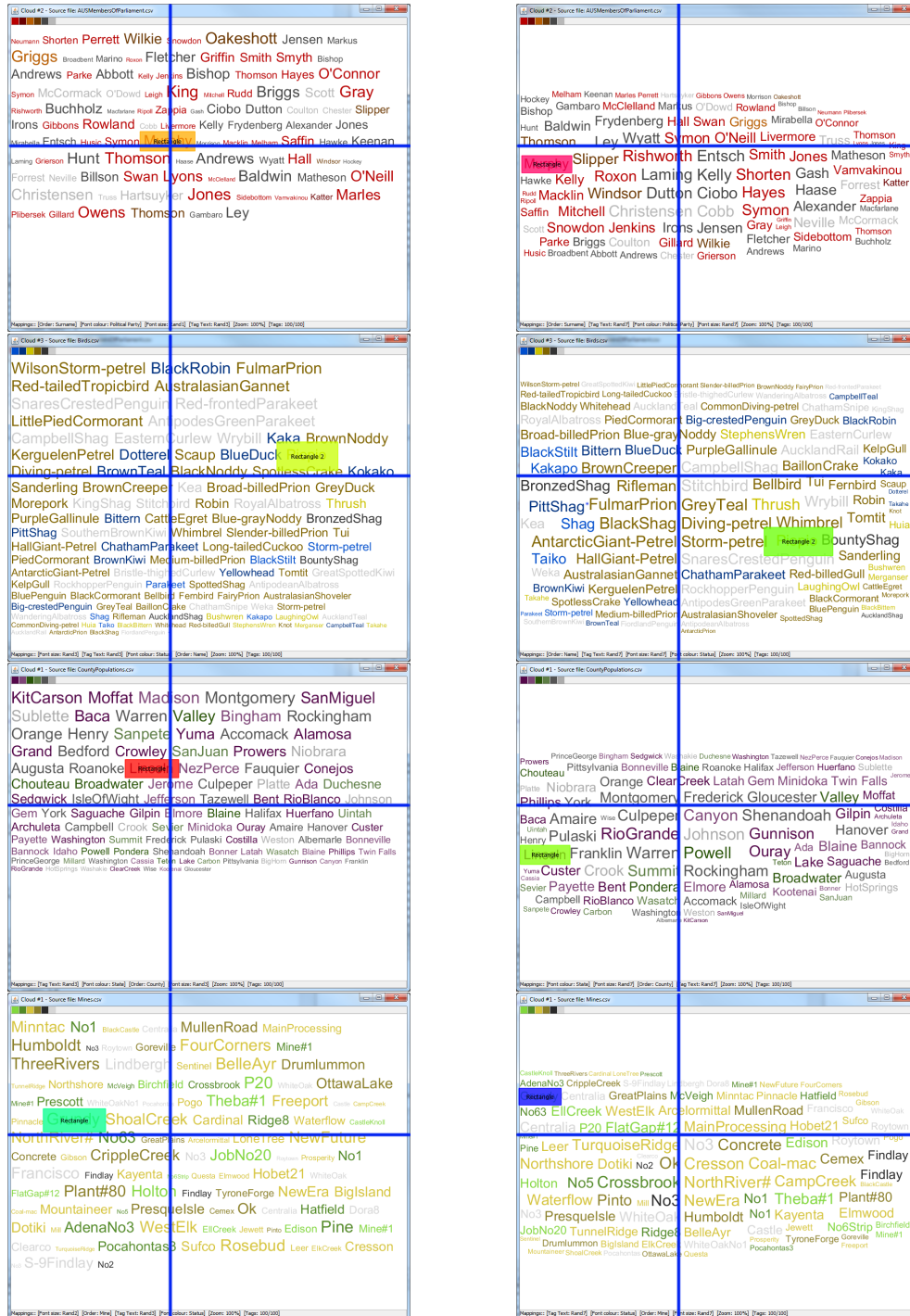


Figure 1: *Experiment one: foreground colour with large target, typewriter and spiral layout*

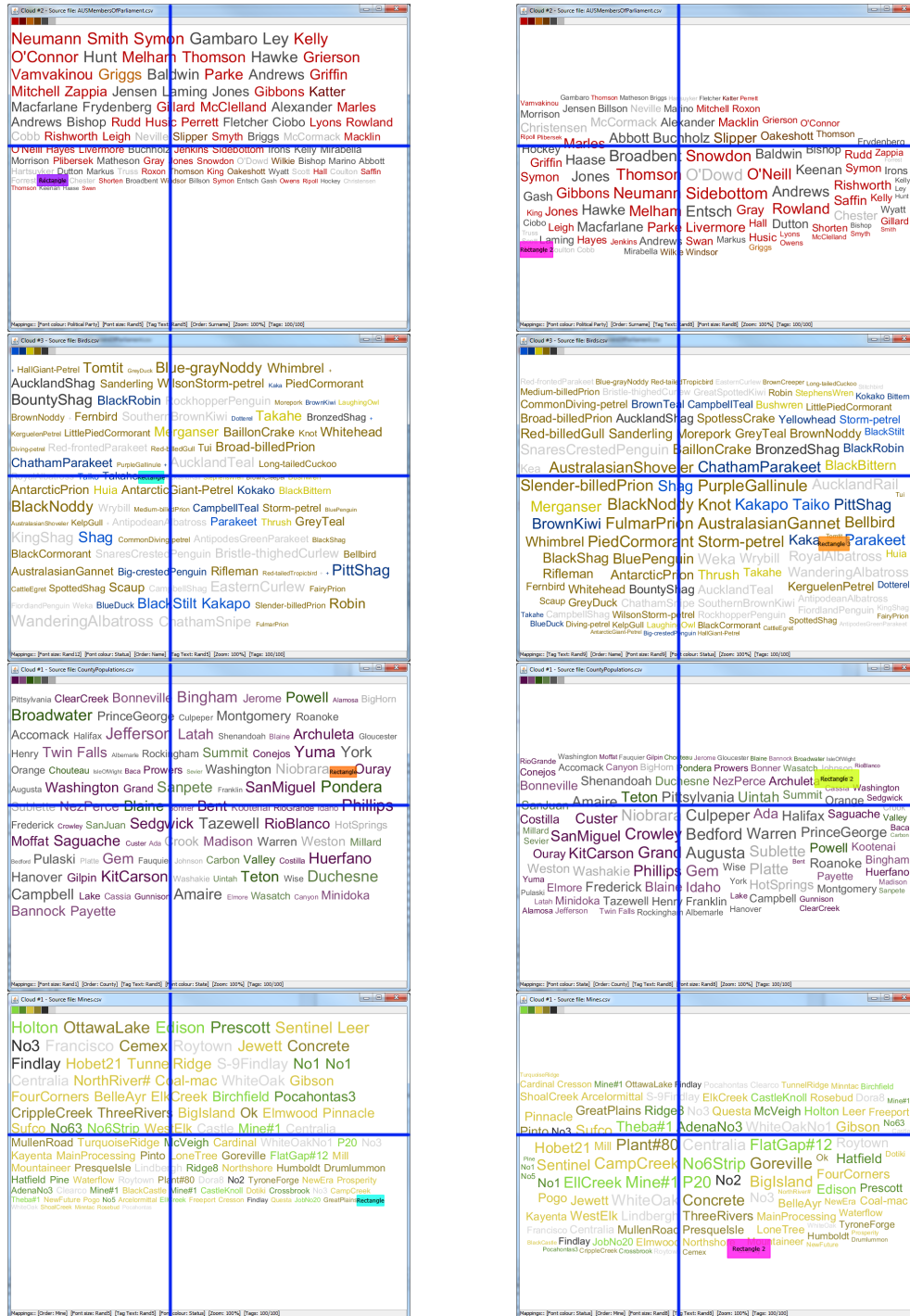


Figure 2: *Experiment one: foreground colour with small target, typewriter and spiral layout*



Figure 3: *Experiment one: background colour with small target, typewriter and spiral layout*

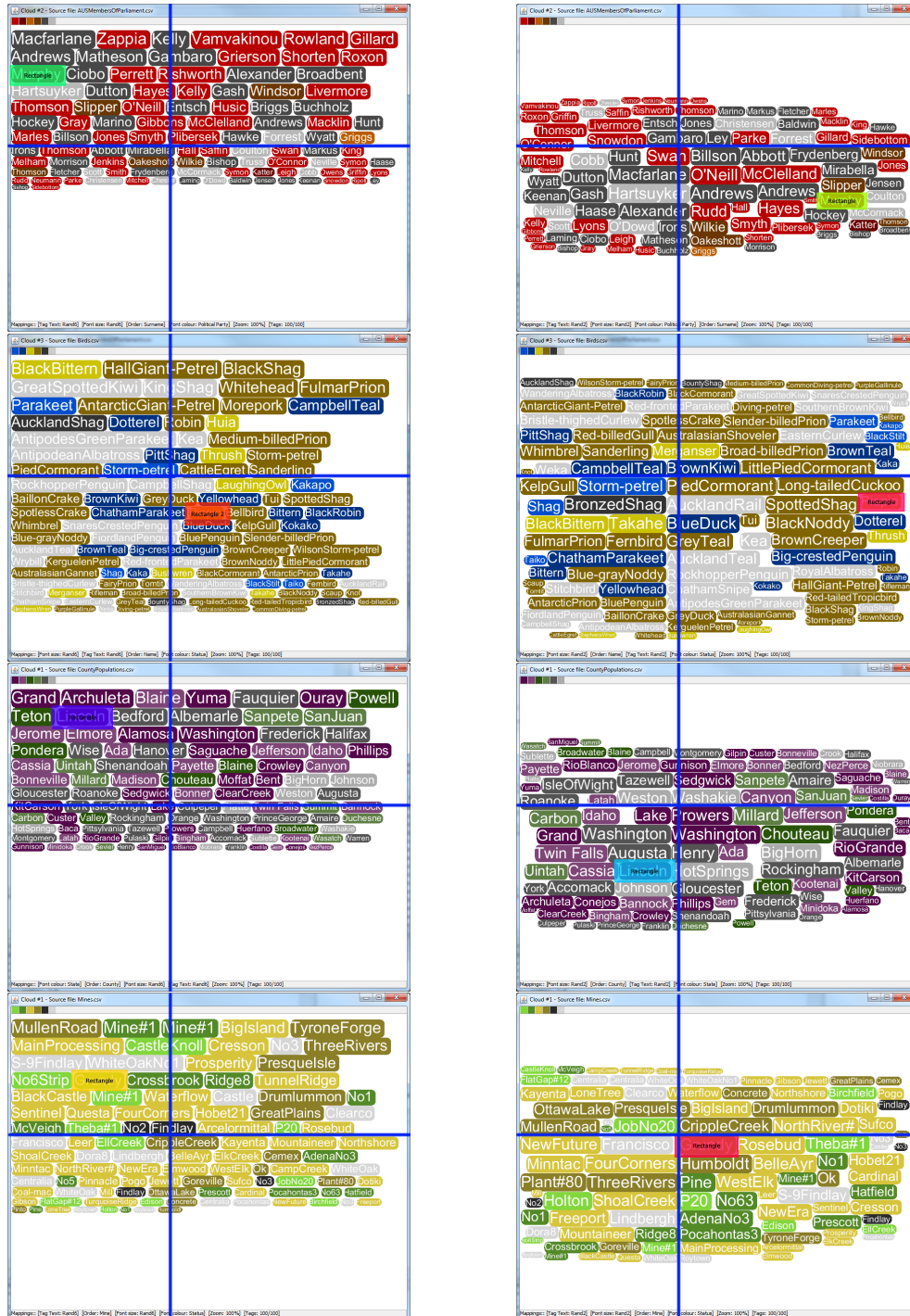


Figure 4: *Experiment one: background colour with small target, typewriter and spiral layout*

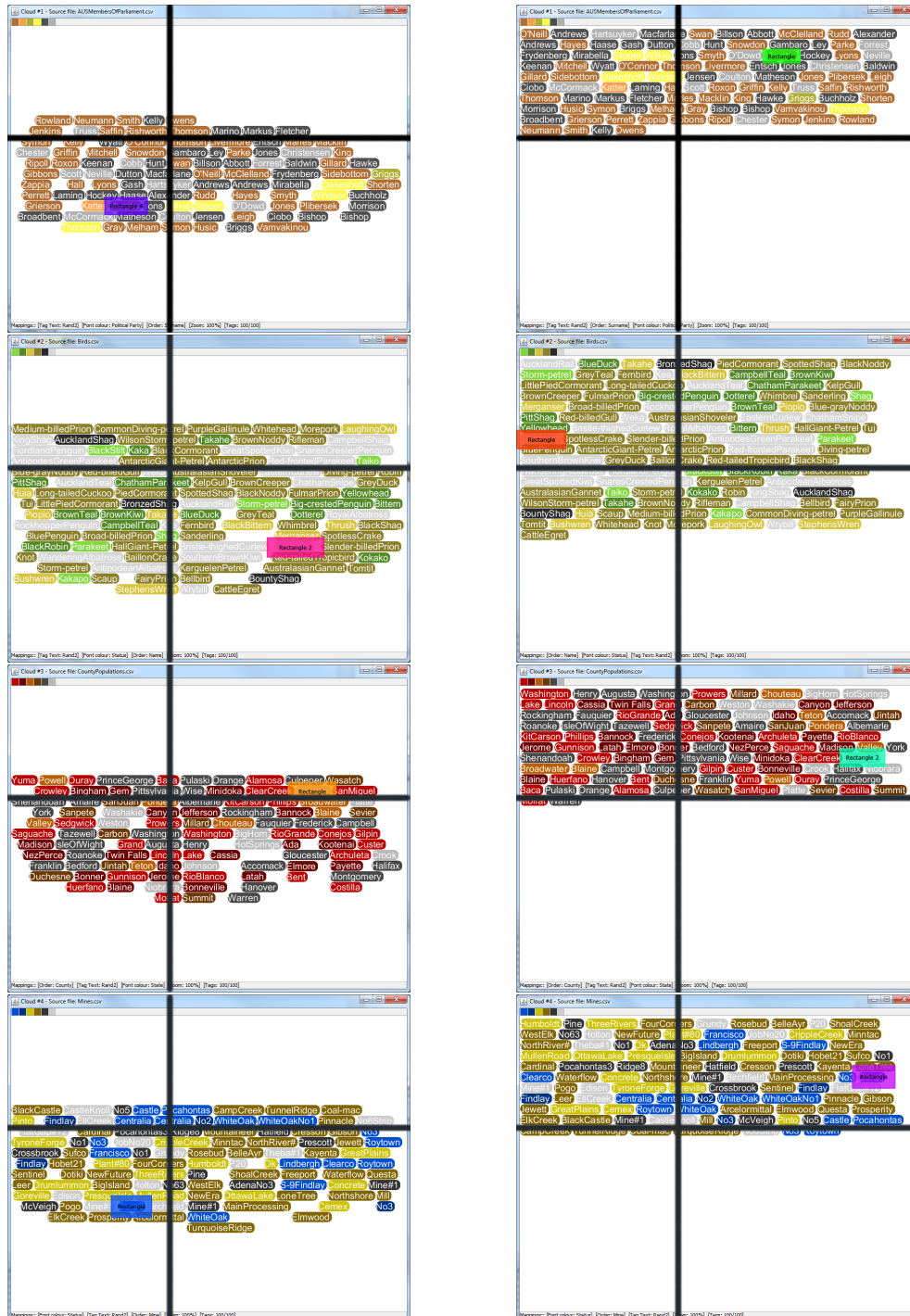


Figure 6: *Experiment two: single colour mapping*



Figure 7: *Experiment two: dual font size and colour mapping*

D Publications

Papers A and B are appended to this thesis.

A Jessica Emerson, Neville Churcher, and Andy Cockburn. Tag clouds for software and information visualisation. In *14th Annual SIGCHI NZ conference on Computer-Human Interaction*. ACM, 2013.

B Jessica Emerson, Neville Churcher, and Chris Deaker. From toy to tool: Extending tag clouds for software and information visualisation. In *Australian Software Engineering Conference*, pages 155–164. IEEE, 2013.

From Toy to Tool: Extending Tag Clouds for Software and Information Visualisation

Jessica Emerson, Neville Churcher and Chris Deaker

Department of Computer Science and Software Engineering, University of Canterbury, Private Bag 4800

Christchurch 8140, New Zealand

Email: neville.churcher@canterbury.ac.nz

Abstract—Software visualisation employs techniques from the more general information visualisation field to help software engineers comprehend and manage the size and complexity of software systems. The scale and complexity of the software engineering domain pose significant challenges and it is important to make effective use of techniques which can be adapted effectively to support tasks in this context. In this paper, we extend significantly the tag cloud concept, transforming it from a simple toy into a powerful tool which can help address challenges inherent in software visualisation. We illustrate our approach with examples drawn from our software engineering research programme and describe TAGGLE a tool which implements our techniques. Our visualisations support developers as they search, filter, browse, explore and act upon data and are a useful addition to the software visualisation tool kit.

I. INTRODUCTION

Despite many advances in theory, tools and practices, scaling software development to cope effectively with systems of ever-increasing current sizes and complexities has continued to be very demanding — as is indicated by the number of IT projects which fail or are significantly compromised. The problem is multi-faceted: more than just technical aspects are involved. Developers need to communicate with their future and former selves: diagrams and visualisations provide convenient *time capsules* for this purpose.

Software visualisation, a sub-field of information visualisation, involves quantities which have no intrinsic geometric representation: computed geometry based on an underlying metaphor underpins individual visualisations. The chosen metaphor has a large influence on the success or otherwise of a visualisation and some general principles have emerged: examples include *exploit human perception strengths*, motivating techniques such as Chernoff faces [1] and geons [2], and *limit the number of data items to be managed concurrently* which leads to the “magic number” 7 ± 2 [3].

There is no single right way to go about designing a visualisation: the overall effectiveness of a metaphor, and the specific techniques used to realise it, depend on factors such as scale, specific tasks and individual variations.

Scale: a visualisation needs to be both tractable computationally and comprehensible effectively by individuals as data set sizes may range over several orders of magnitude [4].

Task: as a user explores a data set the visualisation itself may need to evolve in order to best represent the user’s changing needs as quantities such as time and working set

change. This is particularly evident in visualisation of dynamic systems (e.g. [5]).

Individual: User differences resulting from factors such as age, experience, or gender can be significant. These factors could affect the assumed levels of dexterity, interface complexity and colour blindness respectively. Human perception is also a factor (e.g. we are better at distinguishing size differences than colour differences) to be considered [6].

A large number of visualisation techniques have been developed and applied [7]–[10]. However, there remains the need to explore new techniques, to evaluate their strengths and weaknesses, and to use them effectively to improve the overall effectiveness of visualisations. In particular, we need to consider carefully how to adapt existing techniques to cope not only with the scale and complexity of data sets in the software engineering domain but also with the nature of the tasks performed by software engineers.

We have previously developed tools for specifying, measuring and visualising properties of software systems. Our work has included both “hard” metrics and “soft” heuristics [11]–[13].

We have used a pipeline-based approach in much of our software engineering work [14], [15]. In the visualisation pipeline, data undergoes a series of processes and transformations (initial capture, selection, filtering, geometry computation, ...) leading to the eventual delivery of a visualisation to users. In earlier work we addressed issues pertaining to the design of pipeline-based software visualisations, and illustrated the use of a parallel design pipeline in the design and evolution of visualisations [16]. We use tag clouds as one of many available techniques for presenting the visualisation to the user for analysis, exploration and action.

Tag clouds are a relatively recent addition to the information visualisation toolkit. While they are appealing and appear to have potential applications in information visualisation and software visualisation, these have not been explored fully or applied effectively.

In this paper, we extend the tag cloud concept in order to accommodate information visualisation features including those specifically relevant to software engineering data. We consider some of the issues which characterise software visualisation as well as more general information visualisation contexts. We describe TAGGLE, our prototype implementation, and present examples drawn from our software engineering research. Our work is motivated by the desire to extend tag



Fig. 1. Tag cloud summarising this paper's content

clouds from a simple toy to a powerful tool which is well suited to the visualisation tasks performed by software engineers.

The remainder of the paper is structured as follows. In the next Section, we give a brief overview of tag clouds and their potential for visualisations. Related work is summarised in Section III. We describe our approach, together with the tool we have implemented, in Section IV and illustrate its support for visualisation tasks with a number of examples from the software engineering domain. Some issues relating to scaling are covered in Section V. Sections VI and VII summarise our results, conclusions and ongoing work. Larger versions of figures from the paper are available at <http://www.cosc.canterbury.ac.nz/research/RG/svg/taggle>.

II. EXTENDING TAG CLOUDS FOR VISUALISATION

Tag clouds have become commonplace. The concept of “subconscious files” as weighted lists of keywords occurs in Douglas Coupland’s 1995 novel *Microserfs* [17] but Wordle (<http://www.wordle.net>) really popularised the concept. They are often used to indicate the content of blogs and have become popular in other information retrieval contexts where the content of a document or document collection is of interest. Figure 1 shows a typical example — Wordle’s representation of the text of this paper.

Tags (text labels for data points) are arranged according to some layout strategy. The font sizes used generally correspond to the frequency with which the tags appear in the document under consideration. Conventional tag clouds typically limit themselves to showing the frequency of tags in the underlying data and limit tag numbers (Wordle’s default is 150).

Tag clouds are often regarded as ‘toys’ because of the contexts in which they appear and the simplicity of the basic technique and its implementations. We aim to show that tag clouds can be extended to become a powerful tool alongside other information visualisation techniques and that they can be applied effectively to the software engineering domain.

In previous work we have begun to explore the potential to extend the tag cloud metaphor to enable clouds to be used as an effective visualisation tool [18], [19]. The key idea involves mapping variables in the data set to visual attributes of the tag cloud. In this way, attributes such as colour may be used to convey additional information rather than just function as “decoration.” A key concept in our approach is to support flexible mappings from variables in the data set to visual attributes of corresponding clouds. Similarly, we extend

the basic concept by providing facilities for interacting with clouds.

The models which are the subjects of software visualisations essentially consist of elements, properties of elements and relationships between elements. Users of the visualisations perform a range of navigation, comprehension and exploration tasks involving:

- Gisting or overall impression forming
- Determining the presence/absence of an element, property or relationship
- Studying the distributions of quantities
- Exploring correlations and relationships between quantities
- Discovering new knowledge from the data

Tag clouds can help with each of these.

In the software visualisation domain, data distributions (such as a histogram of LOC) are typically heavily skewed, may contain large numbers of data points, may cover large value ranges and may include outliers [20]). Outliers may actually be the most interesting data points (e.g. a method which is suspiciously complex for its size) and thus can not be ignored.

Techniques for handling large amounts of data may not themselves necessarily provide a good balance of focus and context information. Combining techniques can allow visual indexing and other metaphors to be applied. No single technique is universally ideal: the challenge is to find “horses for courses” using individual techniques in contexts where they are most effective.

We see tag clouds fitting alongside multivariate data alternatives such as kivi charts, treemaps and parallel coordinates [9], [10].

Another key challenge is the focus+context problem, where drilling in to explore detailed properties of specific components leads to the user losing awareness of the relationships with the remainder of the system.

Whatever technique is used, we need to be able to map sufficient variables to be useful in software engineering contexts and to support users in the information analysis tasks they perform.

We have previously considered the design of software visualisations [16] and make use of that approach in this work.

Naïve application of tag clouds to software engineering data sets demonstrate the potential limitations of the technique and the opportunities for improvements.

There are very few existing tag cloud implementations which support tag cloud visualisation of source code or software metrics data. One is the Sonar Software Analysis Platform (<http://www.sonarsource.org> which includes a basic tag cloud component which we used to produce the cloud shown in Figure 2 for Aspectj — an open source project containing 2,342 classes.

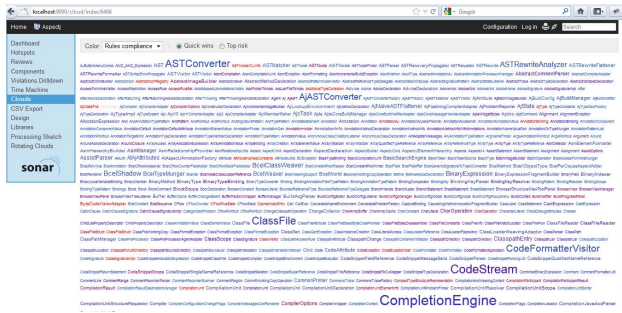


Fig. 2. AspectJ visualised in Sonar

The tag font size is mapped to LOC (Lines of Code) and the tag colour is mapped to a rules compliance metric. The (unqualified) class name is used as the tag identifier — which will prove problematic if class names are not unique.

This cloud is not “nice” — the large number of tags limits readability and there are “rivers” of white space. Even so, it is possible to identify readily the largest few classes and to infer something about the LOC distribution. Individual tags function as hyperlinks to more detailed information about the corresponding classes.

III. RELATED WORK - EXTENSIONS AND EVALUATIONS

Some studies have been done on the evaluation of tag cloud usage but little of this relates directly to the activities specific to our software engineering research. While the general principles have informed our approach, our own empirical studies will be required.

Rivadeneira et al. conducted an early empirical tag cloud evaluation, and proposed a set of user tasks supported by tag clouds [21]. These include searching, browsing, impression forming and recognition/matching. Font size has been discovered to have a strong effect on user perception [21]–[23], although Bateman suggested this effect is influenced by other visual properties. Evaluation of eye-tracking data supports this suggestion [24].

It has been noted that tags in the upper left quadrant of a tag cloud are found more quickly [22] and are better recalled [21]. Analysis of eye-tracking data has shown the upper left quadrant of a tag cloud receives the most attention [24], [25].

Studies [22], [23, for example] have suggested users ‘scan’ tag clouds rather than read them and research considering visual exploration strategies of tag clouds and recorded gaze data has supported this finding [24], [25]. Schrammel et al. also found evidence for two types of user search strategy — chaotic and serial scanning. Users typically switched to serial scanning (performed in a characteristic zig-zag pattern), after a chaotic search pattern did not yield a result.

Hearst and Rosner [26] investigated the motivations behind tag cloud usage. Results from their interviews indicated that users perceive a primary advantage of tag clouds as signallars of the social activity occurring within a system.

Oosterman and Cockburn [27] found that tables were faster and more accurate than tag clouds for some specific

element identification tasks. Others [21], [23], [28] found results indicated ordinary lists perform better than tag clouds for user search tasks, although reported user satisfaction was higher with tag clouds [28].

Users preferred a traditional search interface for specific information retrieval tasks but the tag cloud was preferred where the information seeking was non-specific. They concluded tag clouds were worthwhile as a supplement to traditional interfaces rather than a replacement [29].

Improvements to tag cloud construction and layout have been suggested [30], [31]. Semantically clustered tag clouds for search and recall tasks have also been evaluated [32]. Results indicated that for specific search tasks, semantically clustered tag clouds could improve performance over random layouts, although not alphabetical layouts. Users seem able to more easily find tags in alphabetically ordered clouds [23].

Modified tag clouds have been proposed with various applications. Fujimura et al. presented an algorithm for displaying large datasets in a tag cloud [20]. A layout algorithm to display a hierarchy in a way that captures contextual relationships between tags and promotes navigation has been suggested [33] as have coupling trend charts with word clouds to illustrate temporal content evolutions in a set of documents [34], investigating semantically informed navigation within a tag cloud [35], evaluating tag cloud usage for image retrieval [36] and integrating sparklines into tag clouds to convey trends between multiple tag clouds [37].

Much of this research may be drawn upon when considering the applications of tag clouds or other visualisations in the software engineering domain. For example, the identification of corresponding software engineering tasks to searching, browsing, impression-forming and recognition/matching as classified by Revanedeira et al. [21]. The analysis of eye-tracking data showing user preference for the upper left quadrant of a visualisation is also interesting, since popularly used software diagrams such as UML class diagrams, tend not to use this space extensively.

However, we need to be aware that there are several issues, some of which were noted in Section II, specific to the data sets occurring in software visualisation which must be considered when taking these studies into account.

IV. TAG CLOUDS WITH TAGGLE

In this section we describe the design, implementation and use of TAGGLE. Further detail may be found elsewhere [19]. TAGGLE serves both as a prototype for users and as a platform for ongoing research into layout algorithms and other areas. TAGGLE is written in Java and makes particular use of the Java 2D API.

We illustrate many of the key ideas using examples from the software engineering domain in order to indicate TAGGLE features in context.

Data is maintained in XML files conforming to a DTD appropriate for input to TAGGLE. This approach allows transformation to and from other common formats — such as that used by ggobi (<http://www.ggobi.org>) and other tools — and facilitates our pipeline approach [14]–[16]. TAGGLE’s internal

data structures provide an API allowing direct connection to application data sources. This is particularly useful where the data set is evolving during a TAGGLE session.

Users “design” clouds by selecting values for the parameters which determine layout, mappings and other properties. Settings can be saved and loaded to allow re-use of particular combinations with different data sets. TAGGLE then takes care of the extraction of the required data, application of mappings and rendering of the cloud. Clouds can be saved and translated to other formats, such as SVG, as required.

Users may select tags as targets for subsequent actions (e.g. removal, new cloud creation, ...), obtain more detail about individual tags, observe areas at higher magnification and so on. Depending on the nature of the operation, the cloud may need to be re-computed and re-rendered.

In our work to date, most of our data sets have been taken from our software metrics projects. For example, we might generate a cloud whose labels are class names, font sizes indicate WMC¹, font colours indicate CBO, font transparency indicates proportion of public elements and font style indicates whether the class is abstract or concrete.

We have implemented several layout algorithm families. Tags are placed one at a time, with the selection order (ascending or descending) and other parameters being chosen by the user. Further algorithms can be plugged in as required. Currently implemented algorithms include:

Typewriter: Tags are placed left to right, skipping to a new line when the next tag cannot be placed on the current line.

Spiral: The first tag is placed at the centre of the region, with successive tags being placed around it in a spiral pattern. If the current tag cannot be placed in the first candidate position then several attempts may be required before a suitable location is found. The unsuitable locations will then be used as candidate locations for subsequent tags.

Force-directed: Relationships between tags are used to promote clustering of tags which “belong” together. Each cluster is initially laid out using a spiral layout. A force-directed placement model is used to determine the ultimate layout [15], [39].

Examples of these algorithms in action are presented in §IV and further information is available elsewhere [19]. Videos showing the placement process in more detail may be seen at <http://www.cosc.canterbury.ac.nz/research/RG/svg/taggle>.

Figure 3 shows some typical examples from a session with TAGGLE. The data set contains 75 (public, non-constructor) methods from a Java application.

Users specify *mappings* from variables in the data set to visual attributes of the rendered cloud. Figure 3(g) shows detail of the font size mapping. In this example the font size is determined by the value of statement count (a LOC variant) for each method, mapped via a linear transformation to sizes in the 10–36 point range. A non-linear mapping is available for use when the data values have very large ranges. Similar configuration options are available for font colour, transparency, font family and style.

Figure 3(a) shows a cloud resulting from the mappings $statements \mapsto font\ size$; $cyclomatic\ complexity \mapsto colour$. Colour chips at the top right indicate the legend corresponding to the current colour mapping and current mappings are summarised in the status bar at the bottom of the window (visible in Figures 3(i) and 3(j)).

A typewriter layout with tags ordered by label has been specified. Alphabetical orderings are particularly effective for tasks involving searching for a specific tag (or confirming its absence). In this ordering the variation in font size leads to areas of white space as successive lines are “forced apart” by a large tag.

Figure 3(b) shows the same data with the mappings changed to $statements \mapsto colour$; $cyclomatic\ complexity \mapsto font\ size$. Comparing clouds based on the same data set but with different mappings is straightforward. A feature is provided to allow duplication of a cloud and a new cloud can also be formed from selected tags.

Since we perceive visual attributes differently (e.g. we are generally better at distinguishing variations in size than in colour or transparency and individuals also exhibit variations in perception) the ability to switch mappings conveniently is important for tasks involving correlation or other relationships.

Figures 3(e) and 3(f) show the same mappings with the tags ordered by cyclomatic complexity and LOC respectively. The relative position of a tag in these two clouds allows its position on a corresponding scatter plot to be estimated.

Individual tags may be examined in more detail as required. Figure 3(c) shows the detail of all fields for the tag labelled `next_token` — including those not mapped to any visual attribute.

A region may be magnified as shown in Figure 3(d). We envisage extending this simple facility to a form of bifocal display [40].

Selecting tags, individually or as a region, allows users to:

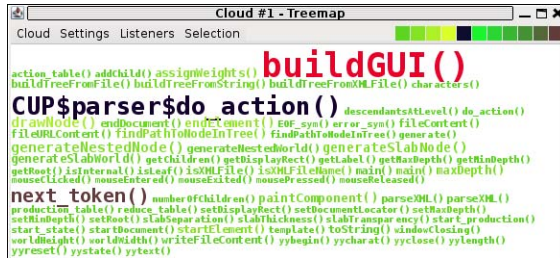
- move the selected tags to a desired position manually
- remove the selected tags from the current cloud
- retain the selected tags and remove the others
- create a new cloud containing the selected tags.

However, it is also useful to be able to filter out tags matching specified criteria. We currently have a simple expression-based filtering mechanism, shown in Figure 3(h). The expressions may involve variables which are not currently mapped, a useful way to explore higher dimensional data sets.

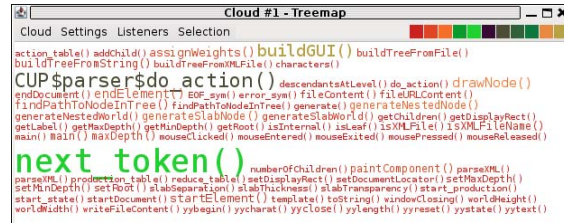
Figures 3(i) and 3(j) show the same data set with a spiral layout selected. These layouts are less suitable for tasks involving locating a specific tag (or confirming its absence) but allow closer packing, avoiding issues of irregular white-space between rows as experienced in typewriter layouts (such as that of Figure 3(b)).

We have found tag clouds to be well suited to tasks such as search, overview and comparison. However, as for many other other techniques with similar strengths (such as SeeSoft [41]) they do not initially appear to be so well suited to tasks

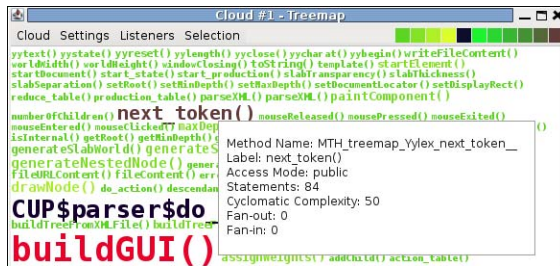
¹WMC and CBO are two of Chidamber and Kemerer’s OO metrics [38]



(a) Ordered by label, no filtering



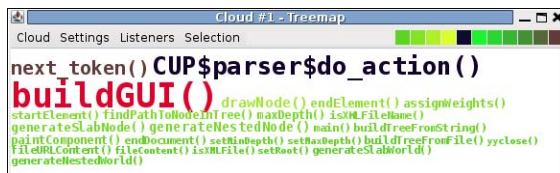
(b) Ordered by label (reverse), CC \rightarrow size, LOC \rightarrow colour



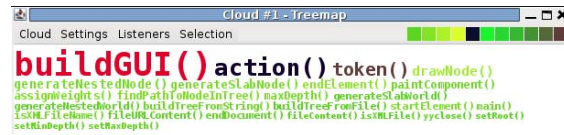
(c) Tag detail



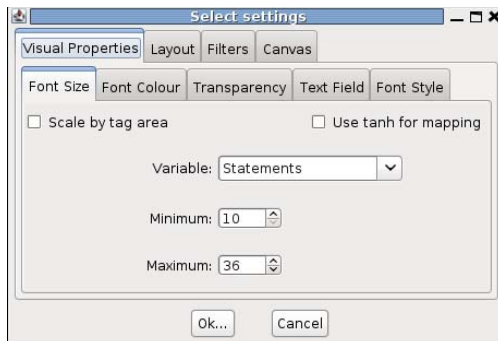
(d) Region zoom



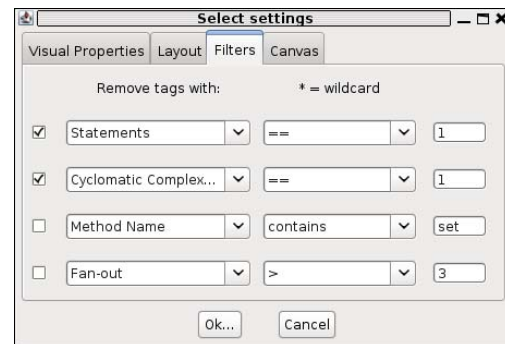
(e) Ordered by CC descending



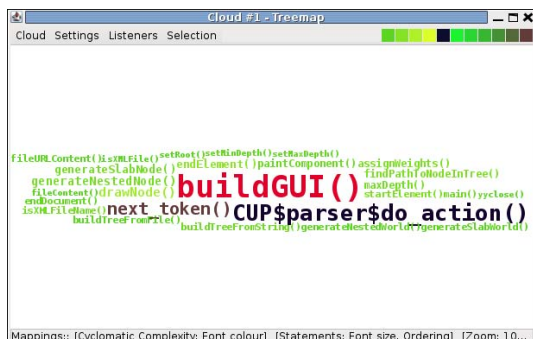
(f) Ordered by LOC descending



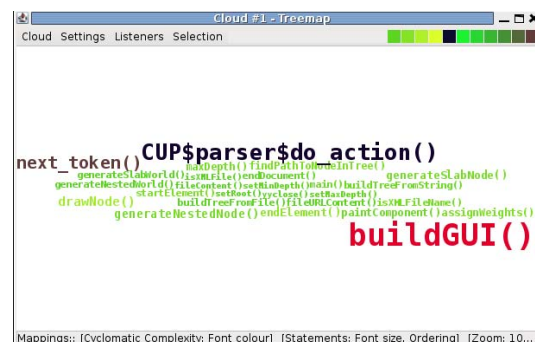
(g) Font size mapping



(h) Filter selection



(i) Spiral, ordered by LOC descending



(j) Spiral ordered by LOC

Fig. 3. Java application examples

acceptable equilibrium. The various factors contributing to the forces may be configured for finer control over the layout using GUI controls.

Figure 4(d) shows a force directed layout. The mappings are *logged hours* \mapsto *font size*; *staff ranking* \mapsto *colour*. Relationship lines indicate group membership and a force layout has been used in order to separate groups more clearly. This approach starts to move a little away from the classical tag cloud in the direction of general graph layout. Nevertheless, the ability to switch layout algorithms, variable mappings and filters with minimal effort adds considerable flexibility to the approach.

Figure 4(e) illustrates the use of highlighting and linking. In Figure 4(e) font size indicates the number of hours logged by each student. The cloud layout is spiral and ordered by group. Some tags have been highlighted (the current binding uses colour to indicate highlighting but custom extensions are provided for).

TAGGLE allows a *highlight listener* to be added. When a tag is selected, the listener highlights the *related* tags. This is another powerful exploration tool, and provides facilities similar to *brushing* as found in tools such as ggobi.

Highlight listeners may be added to *other* clouds. This enables groups of clouds to be used in a manner reminiscent of a scatterplot matrix or other environment where multiple information graphics are available. This is a particularly effective way to explore more than one relationship concurrently. The other clouds are typically created by duplicating the current cloud or from a selected subset of the current cloud's tags. Different mappings will generally be selected in order to explore more relationships.

Figure 4(f) shows the same tags with font size now mapped to grade awarded by staff and no colour mapping. The layout here, a typewriter layout with student names in alphabetical order, makes it easy to find a given student while removing any group clustering.

Highlight listeners have been selected so that when a tag representing a student is selected, the student's group partners are highlighted in both clouds. Figures 4(e) and 4(f) correspond to selecting the tag 'Clouseau' in Figure 4(f). Thus we see that although Figure 4(f) shows that Clouseau has a lower staff grade than his partners, Figure 4(e) suggests that this might be because he has logged fewer hours than his partners.

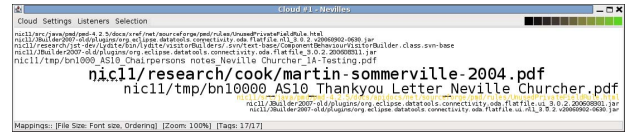
Figure 5 shows another force directed layout containing 66 tags corresponding to classes in a Java application. Font size indicates their WMC metric values and edges indicate inheritance relationships. Linking via highlight listeners to related clouds (such as those of Figure 3) allows exploration.

V. SCALING — PUSHING THE BOUNDARIES

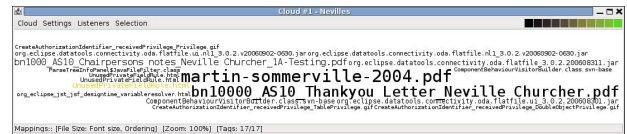
Any visualisation technique comparable to tag clouds for information visualisation purposes will eventually encounter limits as the dimension of the data set (number of variables) and/or the number of data points increases. Sometimes further limitations arise because of user characteristics (e.g. our limited ability to distinguish differences in transparency). In this section we describe how we deal with such issues in TAGGLE.



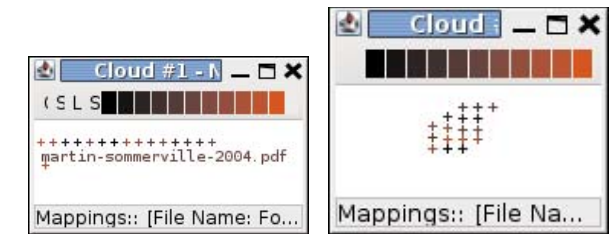
(a) Long tags



(b) Constraining tag length by truncation



(c) Fully truncated



(d) Shrinking window (or adding many tags)

(e) All tags collapsed

Fig. 6. Strategies for coping with structured and oversize tags

Figure 6 illustrates some of the challenges to tag clouds as the boundaries are pushed to more realistically represent data likely to occur in software visualisation applications. These challenges are by no means unique to tag clouds as a visualisation tool, but do need to be addressed if they are to be widely applicable.

One of us (N.C.) wondered how many of his files had names containing the letters of his own name in order. Figure 6 shows the file names returned by the corresponding Unix `find` command with font size mapped to file size.

As can be seen in Figure 6(a), some of the file names are very long. Long tags are problematic. They are clearly hard to place — particularly if they are longer than the width of the tag region. This situation may arise if the tag label is long (as in Figure 6(a)) or where the window size is reduced by the user.

A further difficulty is that when tag lengths vary considerably, a long tag appears to “claim” more real estate in a cloud than it deserves since even if the font size is very small it will still require more space than a shorter tag. We have explored algorithms which constrain tags’ aspect ratios as one way to deal with this though we are not convinced that this is the best solution.

If a tag cannot be placed because it exceeds the width of the available space then it is replaced by a glyph — a ‘+’ in

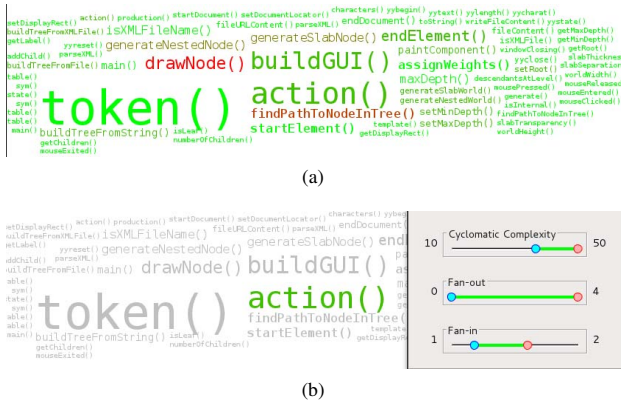


Fig. 7. Dynamic query filtering

Figure 6(a). The details of such collapsed tags may be examined, as is shown for `UnusedPrivateFieldRule.html` in Figure 6(a), in order to decide whether filtering, resizing the window or other actions are appropriate for the task in hand.

In our software visualisation work, long tags tend to be structured implicitly, with a delimiter such as ‘/’ or ‘.’ separating the levels. Examples include file/path names, URIs, fully qualified identifiers and mangled identifiers. We have implemented a feature which allows successive truncations based on the delimiter specified. Figure 6(b) shows the cloud of Figure 6(a) after 4 truncations with the delimiter ‘/’. Note that the tag `UnusedPrivateFieldRule.html`, with the leading `/home/cosc/staff/` having been removed, is now able to be placed (shown highlighted in Figure 6(b)). Figure 6(c) show the fully-truncated version in which all tags are able to be placed.

Figure 6(d) shows the effect on the layout of Figure 6(c) of reducing the window size. The algorithm has reduced the font size in order to place as many tags as possible (only 1 at this point) and the others are all collapsed. The effect of reducing the size even further is shown in Figure 6(c). Even such low level of detail can still be helpful in visualisation tasks: properties such as tag colour remain visible and tags may still be selected and moved.

Figure 7 shows another powerful mechanism for exploring large clouds. We have implemented a form of dynamic query [42]. This allows active filtering based on values of all variables — whether or not they are mapped to a display attribute such as colour or size. The filtering does not change the layout (though there are other facilities to create a new cloud from selected tags) but the filtered tags may be dimmed or removed altogether in order to highlight others. For numeric variables the minimum and maximum values are specified via slide controls and text expressions are used for other fields.

Figure 7(a) shows a cloud of 75 method names from a Java package with mappings *cyclomatic complexity* \mapsto *font size*, *fan-out* \mapsto *colour* in a spiral layout where methods with placement order determined by descending LOC value. Figure 7(b) illustrates the result of limiting the range of the cyclomatic complexity (mapped to font size) and fan-out (unmapped) — only one tag remains and is clearly distinguished from `buildGUI()` which otherwise has very similar visual attributes.

We have found this approach to be sufficiently flexible and powerful to deal with a range of data sets. Ultimately, when limits are encountered we can still gather sufficient information to allow switching to alternative techniques where necessary. This is a common task pattern in information visualisation: one technique (tag clouds) is used to filter and explore higher “big-picture” levels of context and others (such as tables) are used to obtain detail of selected data items.

VI. DISCUSSION & EMPIRICAL ISSUES

Having used TAGGLE on a wide range of data sets, we have formed some general views on the relative merits of tag clouds and our TAGGLE prototype application.

How many tags can be managed effectively? We believe that tag clouds are best suited to tasks where the number of tags to be displayed at any one time is no more than about 50. In practice “right sizing” a cloud involves factors such as:

- filtering to remove tags of currently low relevance.
- tag label choice. A ‘short name’ variable, or coding, can often be chosen if there is no obvious delimiter to use for truncation.
- allowing sufficient real estate (dimensions and aspect ratio) to allow cloud layout algorithms sufficient headroom.

TAGGLE provides us with a platform for experimenting with layout algorithms. It has not been highly optimised, but performance is satisfactory for our purposes — the largest clouds shown in this paper took no more than a few seconds to lay out. However, our current spiral layout algorithms could benefit from re-factoring in order to improve performance for large clouds. Since very large clouds are not particularly comprehensible without filtering or other reduction techniques, such limitations have not proved problematic thus far. The extensible nature of TAGGLE supports the development of new algorithms, visual attribute mappings and interaction mechanisms.

Some studies, such as those mentioned in Section III, of the usability and efficacy of tag clouds have been made. In some cases, only typewriter algorithms are considered. In others, the data sets are artificial or familiar to the users. The variation evident to date means there is no clear standard to compare with. Nevertheless, there are some aspects (e.g. larger tags attract more user attention) where consensus has been achieved and these provide us with guiding principles for our continued work.

While these issues affect the absolute value of tag clouds, we must also consider their merits relative to alternative techniques for a given task. We are interested in issues such as whether users “read” a cloud by scanning left-to-right, top-to-bottom and the effect of the distribution of tag label lengths. These are issues of wider significance. There are also issues specific to software visualisation which we wish to explore.

To supplement our own experiences and anecdotal feedback from our colleagues and students we are currently conducting experiments to shed more light on these issues. A Tobii eye tracker (<http://www.tobii.com>), captures for later analysis data about where the user is looking while performing a task.

Figure 8 shows some indicative results obtained by tracking a user (NC) over a period of approximately 10 seconds as he looked at the cloud shown in Figure 8(a). The tags represent 25 Java classes, with size and colour mapped to metrics from the Chidamber & Kemerer suite.

Figures 8(b)–8(d) show *heatmaps* at 0.5s, 4.9s and 10s respectively. Regions where the user’s attention spends the most time appear in red; green indicates regions given relatively little attention.

Figures 8(e) and 8(f) show *gaze maps* at 0.5s and 10.1s respectively. The circles represent periods where the user’s gaze is focused for a time longer than the current threshold: they are numbered in sequence and the lines show the direction of eye movement between successive attention foci.

Video clips showing the development of these patterns in greater detail are available at <http://www.cosc.canterbury.ac.nz/research/RG/svg/taggle> together with information about the experimental configuration.

VII. CONCLUSION & FUTURE WORK

In this paper we have demonstrated that tag clouds can be extended to include additional information visualisation functionality. This enables tag clouds to be used effectively in software visualisation.

We have described a tool, TAGGLE, which implements our extended tag cloud models. We are encouraged by our experiences thus far, and are continuing to develop our techniques further. One current project involves include tag clouds in standard javadoc HTML pages.

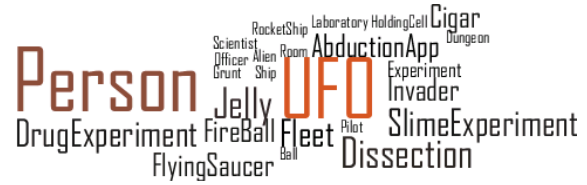
We have used TAGGLE on data from software engineering projects; typical variables are class names, method names, access modes, and software metrics such as lines of code, cyclomatic complexity and the Chidamber & Kemerer suite. Using the mapping mechanism, with its linear and non-linear (tanh) transformations has allowed us to gain insight into our data sets as effectively as other techniques we have used.

As well as being usable in its current form, TAGGLE provides an extensible platform for ongoing exploration of layout algorithms, mapping techniques and other aspects of our software visualisation research programme.

Evaluation is a vital, but challenging, part of visualisation. Our department has recently obtained a Tobii eye tracker (www.tobii.com) and we are currently engaged in empirical studies to help us both quantify the effectiveness of the current implementation but also to help us determine the most appropriate software visualisation tasks to deploy tag clouds on. Such empirical work complements our ongoing feedback from users and we envisage further applications of tag clouds in our own work.

REFERENCES

- [1] H. Chernoff, “The use of faces to represent points in k-dimensional space graphically,” *Journal of the American Statistical Association*, vol. 68, no. 342, pp. 361–368, 1973.
- [2] P. Irani and C. Ware, “Diagramming information structures using 3d perceptual primitives,” *ACM Transactions on Computer-Human Interaction (TOCHI)*, vol. 10, no. 1, pp. 1–19, 2003.



(a) Cloud



(b) Heatmap after 0.5s



(c) Heatmap after 4.9s



(d) Heatmap after 10s



(e) Gaze map after 0.5s



(f) Gaze map after 10.1s

Fig. 8. Tracking the Aliens

- [3] G. A. Miller, "The magical number 7 plus or minus two: Some limits on our capacity for processing information," *Psychological Review*, vol. 63, pp. 81–97, 1957.
- [4] A. Unwin, M. Theus, and H. Hofmann, *Graphics of Large Datasets: Visualizing a Million*. Springer, 2006.
- [5] J. Ashford, N. Churcher, and W. Irwin, "Dynamic visualisation of software state," in *Australasian Computer Science Conference (ACSC 2011)*, Perth, Australia, 2011, pp. 127–136. [Online]. Available: <http://crpit.com/confpapers/CRPITV113Ashford.pdf>
- [6] W. Cleveland, *The Elements of Graphing Data*. Hobart Press, 1994.
- [7] S. Card, J. Mackinlay, and B. Shneiderman, Eds., *Readings in Information Visualisation: Using Vision to Think*. Morgan Kaufman, 1999.
- [8] R. Harris, *Information Graphics: A Comprehensive Illustrated Reference*. Oxford University Press, 1999.
- [9] R. Spence, *Information Visualisation*. Addison-Wesley, 2001.
- [10] C. Ware, *Information Visualization: Perception for Design*, 2nd ed. Morgan Kaufman, 2004.
- [11] W. Irwin and N. Churcher, "Object oriented metrics: Precision tools and configurable visualisations," in *METRICS2003: 9th IEEE Symposium on Software Metrics*, 2003, pp. 112–123.
- [12] B. Neate, W. Irwin, and N. Churcher, "Coderank: A new family of software metrics," in *ASWEC2006: Australian Software Engineering Conference*, 2006, pp. 369–378.
- [13] N. Churcher, S. Frater, C. P. Huynh, and W. Irwin, "Supporting OO design heuristics," in *ASWEC2007: Australian Software Engineering Conference*, 2007, pp. 101–110.
- [14] W. Irwin and N. Churcher, "XML in the visualisation pipeline," in *Visualisation 2001*, ser. Conferences in Research and Practice in Information Technology, vol. 11. Sydney, Australia: ACS, 2002, pp. 59–68.
- [15] N. Churcher, W. Irwin, and C. Cook, "Inhomogeneous force-directed layout algorithms in the visualisation pipeline: From layouts to visualisations," in *InVis.au 2004*, ser. Conferences in Research and Practice in Information Technology, vol. 35, 2004, pp. 43–51.
- [16] N. Churcher and W. Irwin, "Informing the design of pipeline-based software visualisations," in *APVIS2005*, ser. Conferences in Research and Practice in Information Technology, vol. 45. ACS, 2005, pp. 59–68.
- [17] D. Coupland, *Microserfs*. HarperCollins, 1995.
- [18] C. Deaker, L. Pettigrew, N. Churcher, and W. Irwin, "Software visualisation with tag clouds," in *ASWEC 2010 Industry Track Proceedings*, Auckland, New Zealand, Apr. 2010, pp. 129–133.
- [19] C. Deaker, N. Churcher, and W. Irwin, "Tag clouds in software visualisation," Department of Computer Science and Software Engineering, University of Canterbury, Christchurch, New Zealand, Technical Report TR-COSC 01/11, 2011.
- [20] K. Fujimura, S. Fujimura, T. Matsubayashi, T. Yamada, and H. Okuda, "Topigraphy: visualization for large-scale tag clouds," in *Proceedings WWW '08*, 2008, pp. 1087–1088. [Online]. Available: <http://doi.acm.org/10.1145/1367497.1367669>
- [21] A. W. Rivadeneira, D. M. Gruen, M. J. Muller, and D. R. Millen, "Getting our head in the clouds: toward evaluation studies of tagclouds," in *CHI '07*, 2007, pp. 995–998.
- [22] S. Bateman, C. Gutwin, and M. Nacenta, "Seeing things in the clouds: the effect of visual features on tag cloud selections," in *HT '08*, 2008, pp. 193–202.
- [23] M. J. Halvey and M. T. Keane, "An assessment of tag presentation techniques," in *Proc WWW '07*, 2007, pp. 1313–1314.
- [24] S. Lohmann, J. Ziegler, and L. Tetzlaff, "Comparison of tag cloud layouts: Task-related performance and visual exploration," in *INTERACT 2009*, ser. LNCS, vol. 5726. Springer, Aug. 2009, pp. 392–404.
- [25] J. Schrammel, M. Leitner, and M. Tscheligi, "Semantically structured tag clouds: an empirical evaluation of clustered presentation approaches," in *CHI '09*, 2009, pp. 2037–2040. [Online]. Available: <http://doi.acm.org/10.1145/1518701.1519010>
- [26] M. A. Hearst and D. Rosner, "Tag clouds: Data analysis tool or social signaller?" in *Proceedings of HICSS '08*, 2008, pp. 160–170. [Online]. Available: <http://dx.doi.org/10.1109/HICSS.2008.422>
- [27] J. Oosterman and A. Cockburn, "An empirical comparison of tag clouds and tables," in *OZCHI '10*, 2010, pp. 288–295. [Online]. Available: <http://doi.acm.org/10.1145/1952222.1952284>
- [28] B. Y.-L. Kuo, T. Hentrich, B. M. . Good, and M. D. Wilkinson, "Tag clouds for summarizing web search results," in *WWW '07*, 2007, pp. 1203–1204. [Online]. Available: <http://doi.acm.org/10.1145/1242572.1242766>
- [29] J. Sinclair and M. Cardew-Hall, "The folksonomy tag cloud: when is it useful?" *J. Inf. Sci.*, vol. 34, no. 1, pp. 15–29, Feb. 2008. [Online]. Available: <http://dx.doi.org/10.1177/0165551506078083>
- [30] O. Kaser and D. Lemire, "Tag-cloud drawing: Algorithms for cloud visualization," *CoRR*, vol. abs/cs/0703109, 2007.
- [31] C. Seifert, B. Kump, W. Kienreich, G. Granitzer, and M. Granitzer, "On the beauty and usability of tag clouds," in *IV 2008*, 2008, pp. 17–25.
- [32] J. Schrammel, M. Leitner, and M. Tscheligi, "Semantically structured tag clouds: an empirical evaluation of clustered presentation approaches," in *CHI '09*, 2009, pp. 2037–2040.
- [33] K. S. Candan, L. Di Caro, and M. L. Sapino, "Creating tag hierarchies for effective navigation in social media," in *Proc SSM '08*, 2008, pp. 75–82. [Online]. Available: <http://doi.acm.org/10.1145/1458583.1458597>
- [34] W. Cui, Y. Wu, S. Liu, F. Wei, M. Zhou, and H. Qu, "Context preserving dynamic word cloud visualization," in *Pacific Visualization Symposium (PacificVis)*, march 2010, pp. 121–128.
- [35] L. Di Caro, K. S. Candan, and M. L. Sapino, "Using tagflake for condensing navigable tag hierarchies from tag clouds," in *Proc 14th ACM SIGKDD*, ser. KDD '08, 2008, pp. 1069–1072. [Online]. Available: <http://doi.acm.org/10.1145/1401890.1402021>
- [36] J. Callegari and P. Morreale, "Assessment of the utility of tag clouds for faster image retrieval," in *Proc MIR '10*, 2010, pp. 437–440. [Online]. Available: <http://doi.acm.org/10.1145/1743384.1743461>
- [37] B. Lee, N. Riche, A. Karlson, and S. Carpendale, "Sparkclouds: Visualizing trends in tag clouds," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 16, no. 6, pp. 1182–1189, nov.-dec. 2010.
- [38] S. Chidamber and C. Kemerer, "A metrics suite for object oriented design," *Software Engineering, IEEE Transactions on*, vol. 20, no. 6, pp. 476–493, jun 1994.
- [39] P. Eades, "A heuristic for graph drawing," in *Proc. 13th Manitoba Conf. Numerical Mathematics and Computing*, D. S. Meek and G. H. J. v. Rees, Eds. Utilitas Mathematica Publishing, 29 Sep.–1 Oct. 1983.
- [40] Y. K. Leung and M. D. Apperley, "A review and taxonomy of distortion-oriented presentation techniques," *ACM Transactions on Computer-Human Interaction*, vol. 1, no. 2, pp. 126–160, 1994.
- [41] T. Ball and S. Eick, "Software visualization in the large," *IEEE Computer*, vol. 29, no. 4, pp. 33–43, Apr. 1996.
- [42] B. Shneiderman, "Dynamic queries for visual information seeking," *IEEE Softw.*, vol. 11, no. 6, pp. 70–77, Nov. 1994. [Online]. Available: <http://dx.doi.org/10.1109/52.329404>

Tag Clouds for Software and Information Visualisation

Jessica Emerson

Neville Churcher

Andy Cockburn

Department of Computer Science and Software Engineering

University of Canterbury

Christchurch, New Zealand

jessica.emerson@pg.canterbury.ac.nz, {neville.churcher, andy.cockburn}@canterbury.ac.nz

ABSTRACT

We have extended the tag cloud metaphor to allow it to be applied to information and software visualisation. A number of issues, such as wide variation in tag length, have been addressed. We have developed a tool, TAGGLE, which implements our approach. In this paper, we present our visualisation technique and discuss the heuristic evaluation and report preliminary results from user trials employed to evaluate the approach and TAGGLE itself.

Keywords

Visualization, Visualization techniques, Tag Clouds, Information Visualization, Software Visualization

1. INTRODUCTION

A key distinguishing feature of information visualisation is that it involves quantities which do not have intrinsic geometric representations. A visualisation is based on computed geometry which, in turn, is based on one or more underlying metaphors. The effectiveness of an information visualisation technique will be affected by the metaphors that underpin it. Some general principles have become widely accepted. Shneiderman's information seeking mantra *Overview first, zoom and filter, then details-on-demand* is widely cited; *exploit human perception strengths* is embodied in techniques such as Chernoff faces and geons.

We are particularly interested in software visualisation — the application of information visualisation techniques to problems in the software engineering domain. This introduces a number of additional challenges — data is difficult to obtain, highly skewed, has large volumes and ranges, outliers, ... — which we have discussed in previous work [8, 1].

In the next section, we describe the extension of tag clouds to the multivariate visualisation domain and introduce the tool, TAGGLE, we have implemented. Section 3 covers the evaluations we have conducted and our conclusions are presented in Section 4.

2. EXTENDING TAG CLOUDS

Tag clouds are often used to give an overview of the contents of speeches, documents and web sites. Wordle (<http://www.wordle.net>) and other tools have made it easy to create clouds but we believe their full potential for information visualisation has not yet been realised.

In “regular” tag clouds the only attribute which carries information is the font size of tags, typically reflecting the frequency with which the tag occurs in an underlying document or collection. Other elements (layout, colour, bounding shape, ...) are generally only decorative. Our approach is to use properties such as these to allow the representation of more variables. In this way, a cloud becomes a representation of a multivariate data set, supporting users in gisting, searching, knowledge extraction and other tasks.

Tag clouds are particularly attractive in our software visualisation work because text labels are an intrinsic element. In other techniques, such as scatter plots or treemaps, text is problematic.

Our previous software visualisation work has been based on a pipeline approach to visualisation in which data undergoes a sequence of transformations in order to produce a visualisation presented to a user [7, 9, 1]. Some transformations involve operations on the data itself (extraction, selection, aggregation, ...) which deliver the required data in a format appropriate to the visualisation metaphor and subsequent processing stages — we are not concerned with those further in this paper. Other transformations involve sets of mappings from quantities in the pipeline to attributes of the rendered visualisation. For example, a Java class may be represented by a red sphere whose radius is proportional to the number of public methods the class has. Our current work allows such a class to be represented by a tag (its name) whose colour denotes its age and whose size indicates the number of methods it has.

In the case of tag clouds, we can identify a number of such visual attributes of individual tags: text/label, location, size, colour (foreground/background), font (family, style, weight, ...), transparency — even blink rate.

The cloud also has a number of attributes. These include layout — the order in which tags are placed and the algorithm used to determine individual tag locations — as well as the bounding dimensions and background colour.

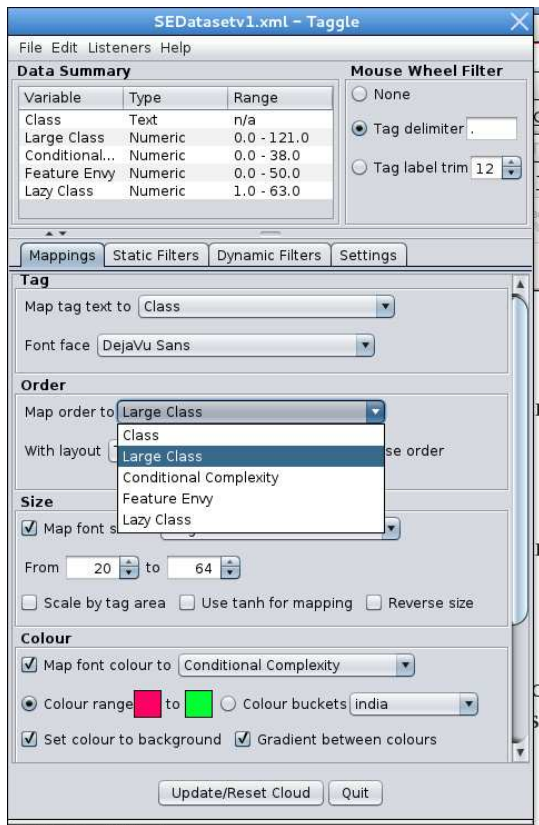


Figure 1: Taggle’s control panel

Our approach, embodied in TAGGLE, involves allowing the user to explore data sets containing values for any number of variables. The user selects particular combinations of mappings from variables to visual attributes (e.g. $\{givenName \mapsto label, age \mapsto fontSize, weight \mapsto fontColour\}$) using TAGGLE’s control panel as shown in Figure 1. These combinations may be saved and reloaded to enable the same analyses to be performed on multiple data sets.

A number of challenging issues arise when extending the concept to the visualisation domain and scaling up to data sets of realistic size. For example, tags with very long labels appear to be more prominent than they deserve to be. If there is considerable variation in the label length then this can be a significant factor.

As the number of tags increases, a point is reached where the display area is full. In order to fit in more tags one could simply reduce font sizes but this is ultimately limited by legibility concerns. Alternatively, labels can be “trimmed” to a maximum length with only the first (or last) parts displayed.

Taking advantage of known structure (e.g. the ‘/’ in file names such as `/usr/local/bin` or the ‘.’ in fully qualified names such as `java.lang.String`) provides a way to reduce label length at natural points.

Filtering (static) allows removing tags with a low Degree Of Interest (DOI) or replacing them with a minimal visual representation. Dynamic filtering allows tags matching a set

of criteria to be dimmed or removed from a cloud.

TAGGLE provides features addressing these and other issues [4].

During a session with TAGGLE, users will typically:

- Create clouds (either from data files or by selecting tags from other clouds)
- Select appropriate mappings from variables in the data set to visual attributes.
- Choose layout algorithms (spiral, typewriter, ...) and layout order.
- Explore clouds: static and/or dynamic filtering; detailed inspection of all quantities associated with a tag (whether or not they are mapped to visual attributes); moving tags manually for detailed comparison; re-mapping to support evolving hypotheses.
- Save clouds and/or mapping sets for display (SVG) or further analysis (XML).

Examples, including videos of TAGGLE in action, are available at <http://www.cosc.canterbury.ac.nz/research/RG/svg/taggle/>.

3. EVALUATION

Our experiences with TAGGLE have satisfied us that there is indeed a rôle for tag clouds in software and information visualisation. However, we now need to identify tasks where tag clouds are particularly effective (and ineffective) in order to deploy them in the most productive contexts. Similarly, we wish to quantify baseline user performance factors and examine the usefulness of TAGGLE’s features.

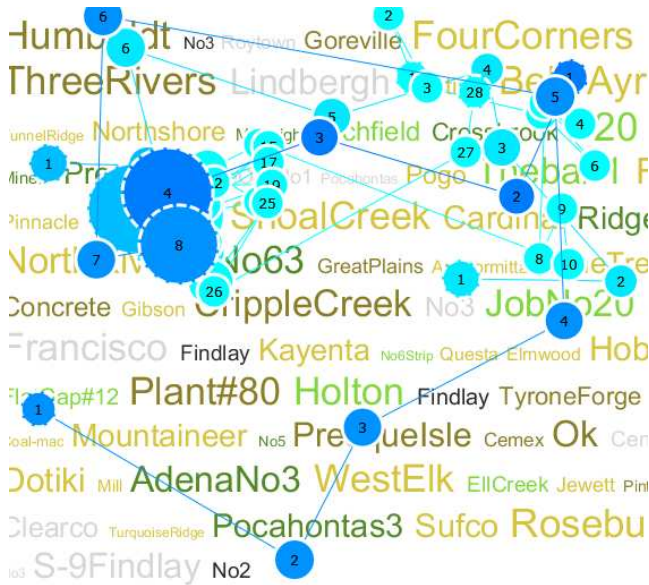
Evaluation in software engineering and visualisation contexts is notoriously challenging for a number of reasons. Typical task complexity and duration is higher than many “normal” interface interactions. Obtaining sufficient numbers of appropriately-skilled subjects is also problematic. Ideally, embedding a tool in a “real” user environment and logging usage would be possible but the impact of doing this with research tools is typically prohibitive.

Consequently, an exhaustive approach based on conventional hypothesis testing alone is impracticable at this stage. Later on, when results from our current studies are available, we expect to identify specific areas where this approach would be helpful.

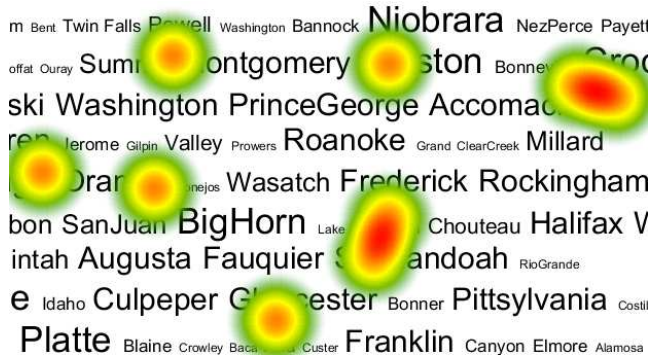
We have approached evaluation in three ways. A systematic mapping study was conducted to collate and review approaches to tag cloud visualisation and evaluation in the literature [3]. This informed the overall approach, including elements covering both whole tool & knowledge discovery and the visualisation technique itself, as well as the detailed experiment design.

We then conducted an heuristic evaluation [10, 11]¹ of TAGGLE [2] using the ten heuristics proposed by Forsell and Jo-

¹See also <http://www.nngroup.com>



(a) Gaze plot, multiple subjects



(b) Heat map, single subject

Figure 2: Sample subject tracking data

hansson [5]. These heuristics cover usability problems for information visualisation systems and are applicable to TAGGLE’s tag cloud visualisations.

The evaluation identified a number of issues, the most significant of which were addressed before the user trials commenced. For example, the data summary table at the upper left of Figure 1 was added because the evaluators wanted a more accessible way to see the available variables and their ranges.

3.1 Trials

For our empirical studies we selected three activities strongly aligned with our visualisation research focus. Our subjects are second and third year computer science and software engineering students. Following initial training in tag clouds and TAGGLE, they performed a series of tasks. Subjects also completed the NASA TLX workload instrument. A Tobii eye tracker and video capture software were used to gather data for subsequent analysis.

3.1.1 Foreground/background colour

Our ability to perceive colour is influenced by a number of factors, the size of the coloured object being one of the main ones. Users of our visualisations will use tag colour as part of the gisting process (e.g. to gain an impression of the distribution of the variable mapped to colour) and other tasks. In correlations between size and colour, users need to be able to confirm statements such as “the smallest tags are the reddest.”

Colours can be compared directly by dragging tags near each other or near the legend. Tag colour may be applied either to the text foreground or background bounding box. The canvas background colour may also be configured by the user.

A $2 \times 2 \times 2$ within-subjects experiment was conducted in which users were presented with combinations of colour placement (foreground, background), target tag size (small, large) and layout (spiral, typewriter) and asked to find a specific tag. Our expectation is that the larger area of colour in the background colour treatment will be associated with shorter completion times and that the difference will be greater for small tag sizes.

Preliminary results (from 5 subjects) appear consistent with our expectations: Tag selection time is faster using background colour ($\mu = 4.92s$) than foreground ($\mu = 6.38s$); tag selection time is faster for large ($\mu = 4.31s$) than small ($\mu = 6.98s$) tags; adding background colour improves selection time more for small ($\mu = 1.9s$) than large ($\mu = 1.0s$) tags. The effect of layout algorithm is currently less clear.

Figure 2(a) shows part of a cloud containing the target tag for a task (foreground tag colour, medium size, typewriter layout). The circles indicate points where a subject’s gaze was focused and the numbers indicate the sequence in which the tags were studied. Results from several users (indicated by colour) are aggregated. Although users take differing times to reach the target, and travel by differing routes, the final gaze positions are co-located and have the longest dwell times.

3.1.2 Parallel mapping

Users are free to establish more mappings than are strictly necessary. A second experiment explores whether mapping both tag colour and tag size to a variable more effective than just mapping either one.

This is a $3 \times 2 \times 2$ within-subjects experiment. Users were presented with combinations of mappings (size, colour, size & colour), target tag size (small, large) and layout (spiral, typewriter) and asked to find a specific tag.

Anecdotal evidence, and our heuristic evaluation, suggests that the *parallel* mapping of both colour and size to a variable will be more effective. Figure 2(b) shows a heat map for a single user performing a task (size only, large tags, typewriter layout). The colours show the aggregated dwell time in each region. In this case, the subject has spent very little time looking at parts of the cloud other than the candidate targets meeting the search criteria but with different labels.

Preliminary results (from 5 participants) have the following

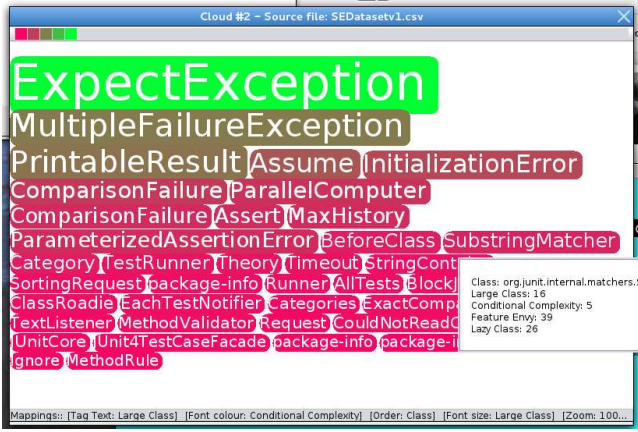


Figure 3: Knowledge discovery task in progress

mean selection times (seconds), suggesting that the parallel mappings improve performance for both layouts.

Mapping	Spiral	Typewriter
size only	4.8	4.5
colour only	4.3	4.6
size & colour	4.0	3.9

3.1.3 Knowledge discovery

Since TAGGLE is intended to be deployed (stand-alone or integrated with IDEs) as part of a software visualisation environment we included a knowledge discovery activity from the SE domain. Such activities are described in the literature[12, for example].

The data set used contains fully qualified class names from a Java program, together with the strengths of a number of code smells [6] — quantities which suggest software has flaws which require refactoring.

Subjects performed a number of benchmark tasks including: finding classes matching smell combination criteria; describing distributions of particular smells; detecting correlated smells and outliers; discussing the similarities and differences between classes.

Figures 1 and 3 show a typical usage of TAGGLE in this activity. Figure 1 shows TAGGLE’s control panel: the user is in the process of mapping the layout order to descending values of Large Class smell and tag colour has been mapped to *Conditional Complexity* smell. The corresponding cloud, shown in Figure 3 illustrates the correlation between these two smells and the skewed distribution of conditional complexity values.

4. CONCLUSIONS

We have extended the tag cloud model to support multi-variate information and software visualisation. TAGGLE incorporates the extended model and allows user to design and explore data sets. It also includes features which address challenges arising from large or complex data sets. A heuristic evaluation has been conducted and we are currently completing user trials which include obtaining eye tracking data. Preliminary results indicate that our tag cloud model

will prove a useful addition to the range of available techniques and we are encouraged to continue its development and to apply it in our software visualisation applications.

5. REFERENCES

- [1] N. Churcher and W. Irwin. Informing the design of pipeline-based software visualisations. In S.-H. Hong, editor, *APVIS2005: Asia-Pacific Symposium on Information Visualisation*, volume 45 of *Conferences in Research and Practice in Information Technology*, pages 59–68, Sydney, Australia, Jan. 2005. ACS.
- [2] J. Emerson. Tag clouds in software visualisation. MSc thesis, Department of Computer Science and Software Engineering, University of Canterbury, 2014. in progress.
- [3] J. Emerson and N. Churcher. Evaluation of tag cloud visualisation: A systematic mapping study. Technical Report TR-COSC 01/13, Department of Computer Science and Software Engineering, University of Canterbury, Christchurch, New Zealand, 2013.
- [4] J. Emerson, N. Churcher, and C. Deaker. From toy to tool: Extending tag clouds for software and information visualisation. In J. Dietrich and J. Noble, editors, *22nd Australian Conference on Software Engineering*, pages 155–164, 4–7 June 2013.
- [5] C. Forsell and J. Johansson. An heuristic set for evaluation in information visualization. In *Proceedings of the International Conference on Advanced Visual Interfaces, AVI ’10*, pages 199–206, New York, NY, USA, 2010. ACM.
- [6] M. Fowler. *Refactoring: Improving the Design of Existing Code*. Addison-Wesley, 1999.
- [7] W. Irwin and N. Churcher. XML in the visualisation pipeline. In D. D. Feng, J. Jin, P. Eades, and H. Yan, editors, *Visualisation 2001*, volume 11 of *Conferences in Research and Practice in Information Technology*, pages 59–68, Sydney, Australia, Apr. 2002. ACS. Selected papers from 2001 Pan-Sydney Workshop on Visual Information Processing.
- [8] W. Irwin and N. Churcher. Object oriented metrics: Precision tools and configurable visualisations. In *METRICS2003: 9th IEEE Symposium on Software Metrics*, pages 112–123, 2003.
- [9] W. Irwin and N. Churcher. Object oriented metrics: Precision tools and configurable visualisations. In *METRICS2003: 9th IEEE Symposium on Software Metrics*, pages 112–123, Sydney, Australia, Sept. 2003. IEEE Press.
- [10] J. Nielsen. Finding usability problems through heuristic evaluation. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 373–380. ACM Press, 1992.
- [11] J. Nielsen and T. K. Landauer. A mathematical model of the finding of usability problems. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 206–213. ACM Press, 1993.
- [12] C. North, P. Saraiya, and K. Duca. A comparison of benchmark task and insight evaluation methods for information visualization. *Information Visualization*, 10(3):162–181, 2011.

References

- OMG Unified Modeling Language (OMG UML), Infrastructure, V2.1.2. Technical report, November 2007. URL <http://www.omg.org/spec/UML/2.1.2/Infrastructure/PDF>. 23
- Sazzadul Alam and Philippe Dugerdil. Evospaces visualization tool: Exploring software architecture in 3d. In *WCRE*, pages 269–270. IEEE Computer Society, 2007. 23
- Frances E. Allen. Control flow analysis. In *Proceedings of a Symposium on Compiler Optimization*, pages 1–19, New York, NY, USA, 1970. ACM. doi: 10.1145/800028.808479. URL <http://doi.acm.org/10.1145/800028.808479>. 23
- Craig Anslow, James Noble, Stuart Marshall, and Ewan Tempero. Visualizing the word structure of java class names. In *OOPSLA Companion '08: Companion to the 23rd ACM SIGPLAN conference on Object-oriented programming systems languages and applications*, pages 777–778, New York, NY, USA, 2008. ACM. 27
- Hidir Aras and Denis Huber. Mobile interaction with geo-notes: A gesture-driven user interface for browsing user-generated content in mobile web applications. In Djamshid Tavangarian, Thomas Kirste, Dirk Timmermann, Ulrike Lucke, and Daniel Versick, editors, *Intelligent Interactive Assistance and Mobile Multimedia Computing*, volume 53 of *Communications in Computer and Information*

REFERENCES

- Science*, pages 25–36. Springer Berlin Heidelberg, 2009. ISBN 978-3-642-10262-2. [82](#), [88](#)
- Avenue. Chernoff faces for evaluations of us judges. http://commons.wikimedia.org/wiki/File:Chernoff_faces_for_evaluations_of_US_judges.svg, April 2010. [20](#)
- Sushil Bajracharya, Joel Ossher, and Cristina Lopes. Searching api usage examples in code repositories with sourcerer api search. In *Proceedings of 2010 ICSE Workshop on Search-driven Development: Users, Infrastructure, Tools and Evaluation*, SUITE '10, pages 5–8, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-962-6. doi: 10.1145/1809175.1809177. URL <http://doi.acm.org/10.1145/1809175.1809177>. [27](#)
- Scott Bateman, Carl Gutwin, and Miguel Nacenta. Seeing things in the clouds: the effect of visual features on tag cloud selections. In *Proceedings of the nineteenth ACM conference on Hypertext and hypermedia*, HT '08, pages 193–202, New York, NY, USA, 2008. ACM. ISBN 978-1-59593-985-2. doi: 10.1145/1379092.1379130. URL <http://doi.acm.org/10.1145/1379092.1379130>. [35](#), [36](#), [37](#), [39](#), [76](#), [82](#), [85](#), [95](#), [134](#), [138](#), [149](#)
- Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, and Dave Thomas. Manifesto for agile software development, 2001. URL <http://www.agilemanifesto.org/>. [12](#)
- Donald Bell. Uml inheritance diagram. http://commons.wikimedia.org/wiki/File:Inheritance_UML_Diagram.jpg, September 2004. [21](#)
- Jacques Bertin. *Semiology of graphics : diagrams, networks, maps / Jacques Bertin ; translated by William J. Berg*. University of Wisconsin Press, Madison, Wis. :, 1983. ISBN 0299090604. [32](#), [36](#), [37](#), [38](#), [39](#), [100](#)
- Thirumalesh Bhat and Nachiappan Nagappan. Evaluating the efficacy of test-driven development: industrial case studies. In *Proceedings of the 2006*

REFERENCES

- ACM/IEEE international symposium on Empirical software engineering*, IS-ESE '06, pages 356–363, New York, NY, USA, 2006. ACM. ISBN 1-59593-218-6. doi: 10.1145/1159733.1159787. URL <http://doi.acm.org/10.1145/1159733.1159787>. 13
- Robert Bringhurst. *The Elements of Typographic Style*. Hartley & Marks Publishers, 2 edition, 2002. URL http://www.amazon.com/Elements-Typographic-Style-Robert-Bringhurst/dp/0881792063/ref=pd_bbs_sr_1/105-5709244-6131609?ie=UTF8&s=books&qid=1188806325&sr=1-1. 32
- Luigi Di Caro, K. Seluk Candan, and Maria Luisa Sapino. Navigating within news collections using tag-flakes. *Journal of Visual Languages & Computing*, 22(2):120 – 139, 2011. ISSN 1045-926X. doi: 10.1016/j.jvlc.2010.11.001. 82
- M. S. T. Carpendale. Considering Visual Variables as a Basis for Information Visualisation. Technical report, University of Calgary, Calgary, AB, 2003. URL http://pharos.cpsc.ucalgary.ca/Dienst/UI/2.0/Describe/ncstr1.ucalgary_cs/2001-693-16. 36, 37
- Herman Chernoff. The Use of Faces to Represent Points in K-Dimensional Space Graphically. *Journal of the American Statistical Association*, 68(342):361–368, 1973. ISSN 01621459. doi: 10.2307/2284077. URL <http://dx.doi.org/10.2307/2284077>. 19
- Neville Churcher, Chris Deaker, and Warwick Irwin. Applications of tag clouds in information and software visualisation. Technical Report TR-COSC 02/11, Department of Computer Science and Software Engineering, University of Canterbury, Christchurch, New Zealand, 2011. 50
- Di Cook. Screenshot of ggobi, showing a parallel coordinate plot. <http://commons.wikimedia.org/wiki/File:Ggobi-flea2.png>, June 2008. 19
- Chris Deaker and Neville Churcher. Taggle technical manual. Technical Report TR-COSC 03/11, Department of Computer Science and Software Engineering, University of Canterbury, Christchurch, New Zealand, 2011. 50, 52, 59

REFERENCES

- Chris Deaker, Neville Churcher, and Warwick Irwin. Tag clouds in software visualisation. Technical Report TR-COSC 01/11, Department of Computer Science and Software Engineering, University of Canterbury, Christchurch, New Zealand, 2011. [50](#), [52](#), [59](#)
- Stephen G. Eick, Joseph L. Steffen, and Eric E. Sumner, Jr. Seesoft-a tool for visualizing line oriented software statistics. *IEEE Trans. Softw. Eng.*, 18(11): 957–968, November 1992. ISSN 0098-5589. [24](#)
- Eva Van Emden and Leon Moonen. Java quality assurance by detecting code smells. In *in Proceedings of the 9th Working Conference on Reverse Engineering. IEEE Computer*, pages 97–107. Society Press, 2002. [23](#)
- Jonathan Feinberg. Wordle. In Julie Steele and Noah Iliinsky, editors, *Beautiful Visualization: Looking at Data through the Eyes of Experts*, chapter 3, pages 37–58. O’Reilly Media, 2010. [26](#)
- Norman E. Fenton and Shari Lawrence Pfleeger. *Software Metrics: A Rigorous and Practical Approach*. PWS Publishing Co, 2nd edition edition, 1998. [14](#)
- Camilla Forsell and Jimmy Johansson. An heuristic set for evaluation in information visualization. In *Proceedings of the International Conference on Advanced Visual Interfaces, AVI ’10*, pages 199–206, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0076-6. doi: 10.1145/1842993.1843029. URL <http://doi.acm.org/10.1145/1842993.1843029>. [100](#)
- Martin Fowler. *Refactoring: Improving the Design of Existing Code*. Addison-Wesley, 1999. [13](#), [14](#), [16](#)
- Frothy. Plot of s&p composite real price index, earnings, dividends, and interest rates. http://en.wikipedia.org/wiki/Robert_J._Shiller, February 2008. [16](#)
- Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1994. [13](#)

REFERENCES

- Diego Gomez-Aguilar, Miguel Conde-Gonzalez, Roberto Theron, and Francisco Garcia-Peñalvo. Supporting moodle-based lesson through visual analysis. In *Proceedings of the 13th IFIP TC 13 international conference on Human-computer interaction - Volume Part IV*, INTERACT'11, pages 604–607, Berlin, Heidelberg, 2011. Springer-Verlag. ISBN 978-3-642-23767-6. URL <http://dl.acm.org/citation.cfm?id=2042283.2042387>. 82
- Martin J. Halvey and Mark T. Keane. An assessment of tag presentation techniques. In *Proceedings of the 16th international conference on World Wide Web*, WWW '07, pages 1313–1314, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-654-7. doi: 10.1145/1242572.1242826. URL <http://doi.acm.org/10.1145/1242572.1242826>. 35, 36, 45, 76, 85, 95, 138
- S. G. Hart and L. E. Stavenland. Development of NASA-TLX (Task Load Index): Results of empirical and theoretical research. In P. A. Hancock and N. Meshkati, editors, *Human Mental Workload*, chapter 7, pages 139–183. Elsevier, 1988. URL http://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20000004342_1999205624.pdf. 155, 167
- U.S. Department Health and Services. *Research-Based Web Design & Usability Guidelines: Current Research-Based Guidelines on Web Design and Usability Issues*. U.S. Government Printing Office, Washington, D.C., 2006 edition edition, 2006. ISBN 0-16-076270-7. URL <http://www.usability.gov/pdfs/guidelines.html>. 36, 37, 55, 56
- Marti A. Hearst and Daniela Rosner. Tag clouds: Data analysis tool or social signaller? In *Proceedings of the Proceedings of the 41st Annual Hawaii International Conference on System Sciences*, HICSS '08, pages 160–, Washington, DC, USA, 2008. IEEE Computer Society. ISBN 0-7695-3075-8. doi: 10.1109/HICSS.2008.422. URL <http://dx.doi.org/10.1109/HICSS.2008.422>. 27, 76
- Brian Henderson-Sellers. *Object-Oriented Metrics: Measures of Complexity*. Prentice Hall, 1995. 14
- W. Irwin and N. Churcher. Object oriented metrics: precision tools and configurable visualisations. In *Software Metrics Symposium, 2003. Proceedings.*

REFERENCES

- Ninth International*, pages 112 – 123, sept. 2003. doi: 10.1109/METRIC.2003.1232460. 23, 51
- T. J. Jankun-Kelly, David Wilson, Andrew S. Stamps, Josh Franck, Jeffery Carver, and J. Edward Swan II. Visual analysis for textual relationships in digital forensic evidence. *Information Visualization*, 10(2):134–144, 2011. 88
- Owen Kaser and Daniel Lemire. Tag-cloud drawing: Algorithms for cloud visualization. *CoRR*, abs/cs/0703109, 2007. 76, 85, 95
- Daniel A. Keim and HP Kriegel. Visualization techniques for mining large databases: A comparison. *IEEE Transactions on Knowledge and Data Engineering*, 8:923–938, 1996. 18
- Thi Nhu Quynh Kim, K. Razikin, Dion Hoe-Lian Goh, Yin Leng Theng, Quang Minh Nguyen, Ee-Peng Lim, Aixin Sun, Chew Hung Chang, and K. Chatterjea. Exploring hierarchically organized georeferenced multimedia annotations in the mobitop system. In *Information Technology: New Generations, 2009. ITNG '09. Sixth International Conference on*, pages 1355 –1360, april 2009. doi: 10.1109/ITNG.2009.330. 82
- J H Krueger. Anaglyph stereo volume rendered image of the visible human male ct data set using imagevis3d. <http://en.wikipedia.org/wiki/File:VisHumanCT-Anaglyph-2.png>, May 2009. 17
- Adrian Kuhn and Mirko Stocker. Codetimeline: storytelling with versioning data. In *Proceedings of the 2012 International Conference on Software Engineering*, ICSE 2012, pages 1333–1336, Piscataway, NJ, USA, 2012. IEEE Press. ISBN 978-1-4673-1067-3. URL <http://dl.acm.org/citation.cfm?id=2337223.2337410>. 27
- Byron Y-L Kuo, Thomas Hentrich, Benjamin M . Good, and Mark D. Wilkinson. Tag clouds for summarizing web search results. In *Proceedings of the 16th international conference on World Wide Web*, WWW '07, pages 1203–1204, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-654-7. doi: 10.1145/1242572.1242766. URL <http://doi.acm.org/10.1145/1242572.1242766>. 27, 45, 76

REFERENCES

- Christopher Kurtz. Code gestalt: a software visualization tool for human beings. In *CHI '11 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '11, pages 929–934, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0268-5. 27, 82, 88
- Lawrence Livermore National Laboratory. Scientific visualization of an extremely large simulation of a rayleigh-taylor instability problem. http://it.wikipedia.org/wiki/File:Rayleigh-Taylor_instability.jpg, September 2009. 17
- Heidi Lam, Enrico Bertini, Petra Isenberg, Catherine Plaisant, and Sheelagh Carpendale. Empirical studies in information visualization: Seven scenarios. *IEEE Transactions on Visualization and Computer Graphics*, 18:1520–1536, 2012. ISSN 1077-2626. doi: <http://doi.ieeecomputersociety.org/10.1109/TVCG.2011.279>. 73, 79, 84, 88, 91, 97
- Lars. Tube map, london. <http://www.flickr.com/photos/solsken/692148358/>, June 2007. 16
- Steffen Lohmann, Jürgen Ziegler, and Lena Tetzlaff. Comparison of tag cloud layouts: Task-related performance and visual exploration. In *Proceedings of the 12th IFIP TC 13 International Conference on Human-Computer Interaction: Part I*, INTERACT '09, pages 392–404, Berlin, Heidelberg, 2009. Springer-Verlag. ISBN 978-3-642-03654-5. doi: 10.1007/978-3-642-03655-2_43. URL http://dx.doi.org/10.1007/978-3-642-03655-2_43. 35, 36, 76, 128, 138
- R. D. Luce. *Response times: Their role in inferring elementary mental operations*. Oxford University Press, New York, 1986. 132, 147
- R.G. Luzzardi, C.M.D.S Freitas, R.A. Cava, G.D Duarte, and M.H.S. Vasconcelos. An extended set of ergonomic criteria for information visualization. In *In Proc. IASTED. International Conference of Computer Graphics and Imaging*, pages 236–241, 2004. 100
- Emerson Murphy-Hill and Andrew P. Black. An interactive ambient visualization for code smells. In *Proceedings of the 5th international symposium on Software*

REFERENCES

- visualization*, SOFTVIS '10, pages 5–14, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0028-5. doi: 10.1145/1879211.1879216. URL <http://doi.acm.org/10.1145/1879211.1879216>. 23
- NASA. Synoptic chart of weather in usa: Source nasa. http://commons.wikimedia.org/wiki/File:Synoptic_weather.gif, March 2006. 16
- Jakob Nielsen. Finding usability problems through heuristic evaluation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, page pages 373380. ACM Press, 1992. 99, 100
- L. O’Grady, C. N. Wathen, J. Charnaw-Burger, L. Betel, A. Shachak, R. Luke, S. Hockema, and A. R. Jadad. The use of tags and tag clouds to discern credible content in online health message forums. *International journal of medical informatics*, 81(1):36–44, 2012. URL www.scopus.com. 88
- Anneli Olsen, Linnea Smolentzov, and Tommy Strandvall. Comparing different eye tracking cues when using the retrospective think aloud method in usability testing. In *Proceedings of the 24th BCS Interaction Specialist Group Conference*, BCS '10, pages 45–53, Swinton, UK, UK, 2010. British Computer Society. ISBN 978-1-78017-130-2. URL <http://dl.acm.org/citation.cfm?id=2146303.2146310>. 96
- Josh Oosterman and Andy Cockburn. An empirical comparison of tag clouds and tables. In *Proceedings of the 22nd Conference of the Computer-Human Interaction Special Interest Group of Australia on Computer-Human Interaction*, OZCHI '10, pages 288–295, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0502-0. doi: 10.1145/1952222.1952284. URL <http://doi.acm.org/10.1145/1952222.1952284>. 45, 76, 95
- OpenClips. Road signs, bicycle, pedestrian walking. <http://pixabay.com/en/road-signs-bicycle-cycle-bike-151608/>, October 2013. 16
- Evan M. Palmer, Todd S. Horowitz, Antonio Torralba, and Jeremy M. Wolfe. What are the shapes of response time distributions in visual search? *Journal of experimental psychology. Human perception and performance*, 37(1):58–71,

REFERENCES

- February 2011. ISSN 1939-1277. doi: 10.1037/a0020747. URL <http://dx.doi.org/10.1037/a0020747>. 132, 147
- Catherine Plaisant. The challenge of information visualization evaluation. In *Proceedings of the Working Conference on Advanced Visual Interfaces, AVI '04*, pages 109–116, New York, NY, USA, 2004. ACM. ISBN 1-58113-867-9. doi: 10.1145/989863.989880. URL <http://doi.acm.org/10.1145/989863.989880>. 72
- Adam M. Preston, John R. Hayes, Yu-Chi Tai, and James E. Sheedy. Emphasis techniques in presentations: Effectiveness and recall. *Optometry (St. Louis, Mo.)*, 81:299, 2010. 37, 39, 40
- S. P. Reiss. The paradox of software visualization. In *Proceedings of the 3rd IEEE International Workshop on Visualizing Software for Understanding and Analysis, VISSOFT '05*, pages 19–, Washington, DC, USA, 2005. IEEE Computer Society. ISBN 0-7803-9540-9. doi: 10.1109/VISSOFT.2005.1684306. URL <http://dx.doi.org/10.1109/VISSOFT.2005.1684306>. 2, 21, 27
- A. W. Rivadeneira, Daniel M. Gruen, Michael J. Muller, and David R. Millen. Getting our head in the clouds: toward evaluation studies of tagclouds. In *Proceedings of the SIGCHI conference on Human factors in computing systems, CHI '07*, pages 995–998, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-593-9. doi: 10.1145/1240624.1240775. URL <http://doi.acm.org/10.1145/1240624.1240775>. 35, 45, 76, 82, 95
- Johann Schrammel, Stephanie Deutsch, and Manfred Tscheligi. Visual search strategies of tag clouds - results from an eyetracking study. In *Proceedings of the 12th IFIP TC 13 International Conference on Human-Computer Interaction: Part II, INTERACT '09*, pages 819–831, Berlin, Heidelberg, 2009a. Springer-Verlag. ISBN 978-3-642-03657-6. doi: 10.1007/978-3-642-03658-3_85. URL http://dx.doi.org/10.1007/978-3-642-03658-3_85. 35, 36, 76, 127
- Johann Schrammel, Michael Leitner, and Manfred Tscheligi. Semantically structured tag clouds: an empirical evaluation of clustered presentation approaches. In *Proceedings of the 27th international conference on Human fac-*

REFERENCES

- tors in computing systems, CHI '09, pages 2037–2040, New York, NY, USA, 2009b. ACM. ISBN 978-1-60558-246-7. doi: 10.1145/1518701.1519010. URL <http://doi.acm.org/10.1145/1518701.1519010>. 35, 76, 95
- C. Seifert, B. Kump, W. Kienreich, G. Granitzer, and M. Granitzer. On the beauty and usability of tag clouds. In *Information Visualisation, 2008. IV '08. 12th International Conference*, pages 17 –25, july 2008. doi: 10.1109/IV.2008.89. 76
- Ben Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *Proceedings of the 1996 IEEE Symposium on Visual Languages*, VL '96, pages 336–, Washington, DC, USA, 1996. IEEE Computer Society. ISBN 0-8186-7508-X. URL <http://dl.acm.org/citation.cfm?id=832277.834354>. 27, 41, 62, 100
- Ben Shneiderman. Treemaps for space-constrained visualization of hierarchies. Web document, 2009. 24
- Yedendra Babu Shrinivasan, David Gotz, and Jie Lu. Connecting the dots in visual analysis. In *IEEE VAST*, pages 123–130. IEEE, 2009. 82
- James Sinclair and Michael Cardew-Hall. The folksonomy tag cloud: when is it useful? *J. Inf. Sci.*, 34(1):15–29, February 2008. ISSN 0165-5515. doi: 10.1177/0165551506078083. URL <http://dx.doi.org/10.1177/0165551506078083>. 45, 76
- Dimitrios Skoutas and Mohammad Alrifai. Tag clouds revisited. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, CIKM '11, pages 221–230, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0717-8. 85
- Robert Spence. *Information Visualization: Design for Interaction*. Prentice Hall, 2nd Edition 2007. 16
- Michael Stroeck. A 3d visualization of a caffeine molecule. http://commons.wikimedia.org/wiki/File:Caffeine_Molecule.png, January 2006. 17

REFERENCES

- Christopher M. B. Taylor and Malcolm Munro. Revision towers. In *VISSOFT*, pages 43–50. IEEE Computer Society, 2002. ISBN 0-7695-1662-9. [24](#)
- Edward R. Tufte. *Envisioning Information*. Graphics Press, Cheshire, CT, 1990. ISBN 978-0-9613921-1-6. [100](#)
- UCRL. Terrain rendering. urlhttp://en.wikipedia.org/wiki/File:Terrain_rendering.jpg, November 2007. [17](#)
- Manuela Waldner, Johann Schrammel, Michael Klein, Katrín Kristjánsdóttir, Dominik Unger, and Manfred Tscheligi. Facetclouds: Exploring tag clouds for multi-dimensional data. In *Proceedings of the 2013 Graphics Interface Conference*, GI '13, pages 17–24, Toronto, Ont., Canada, Canada, 2013. Canadian Information Processing Society. ISBN 978-1-4822-1680-6. URL <http://dl.acm.org/citation.cfm?id=2532129.2532134>. [37](#), [39](#), [56](#)
- Colin Ware. *Information Visualization: Perception for Design*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004. ISBN 1558608192. [16](#), [38](#), [39](#), [100](#), [127](#)
- Mathew J. Wilson and Max L. Wilson. Tag clouds and keyword clouds: evaluating zero-interaction benefits. In Desney S. Tan, Saleema Amershi, Bo Begole, Wendy A. Kellogg, and Manas Tungare, editors, *Proceedings of the International Conference on Human Factors in Computing Systems, CHI 2011, Extended Abstracts Volume, Vancouver, BC, Canada, May 7-12, 2011*, pages 2383–2388. ACM, 2011. ISBN 978-1-4503-0268-5. doi: <http://doi.acm.org/10.1145/1979742.1979913>. [85](#)
- Jeremy M. Wolfe and Todd S. Horowitz. Opinion: What attributes guide the deployment of visual attention and how do they do it? *Nature Reviews Neuroscience*, 5(6):495–501, June 2004. ISSN 1471-003X. doi: [10.1038/nrn1411](http://dx.doi.org/10.1038/nrn1411). URL <http://dx.doi.org/10.1038/nrn1411>. [18](#)
- Takehiro Yamamoto, Satoshi Nakamura, and Katsumi Tanaka. Termcloud for enhancing web search. In Gottfried Vossen, Darrell D.E. Long, and Jeffrey Xu Yu, editors, *Web Information Systems Engineering - WISE 2009*, volume 5802 of

REFERENCES

- Lecture Notes in Computer Science*, pages 159–166. Springer Berlin Heidelberg, 2009. ISBN 978-3-642-04408-3. [85](#)
- T. Van Zandt. How to fit a response time distribution. *Psychonomic Bulletin & Review*, 7(3):424–465, 2000. [132](#), [147](#)
- Torre Zuk and Sheelagh Carpendale. Theoretical analysis of uncertainty visualizations. In *Proc. of SPIE-IS&T Electronic Imaging*, pages 66–79, 2006. [100](#)